Ю.Р. Мухина

ВЕБ-ТЕХНОЛОГИИ: ОСНОВЫ ВЕРСТКИ САЙТОВ

ОУ ВО «Южно-Уральский технологический университет»

Ю.Р. Мухина

ВЕБ-ТЕХНОЛОГИИ: ОСНОВЫ ВЕРСТКИ САЙТОВ

Челябинск

2022

УДК 004.55
ББК 16.263.4
М92

Автор: Ю.Р. Мухина – кандидат педагогических наук, доцент кафедры «Техника и технологии» ОУ ВО «Южно-Уральский технологический университет»

Рецензенты:
Гисс Е.И. – к.п.н., доцент, зав. кафедрой математики, информатики и естественнонаучных дисциплин РАНХиГС, Челябинский филиал;
Овсяницкая Л.Ю. – к.т.н., доцент, заведующий кафедрой математики и информатики ЧОУВО «Международный Институт Дизайна и Сервиса»;

Мухина, Ю.Р.
Веб-технологии: основы верстки сайтов: Учебное пособие [Электронный ресурс] / Ю.Р. Мухина. – Челябинск: ОУ ВО «Южно-Уральский технологический университет», 2022. – Режим доступа: https://www.inueco.ru/rio/2022/978-5-6047814-5-6.pdf, свободный. – Загл. с экрана.– 154 с.

ISBN 978-5-6047814-5-6

Учебное пособие включает в себя теоретический материал, методические рекомендации по выполнению практических работ, перечень вопросов для самопроверки, перечень вопросов для углубленного изучения темы, задания для выполнения проектов, списки рекомендуемой литературы и ссылки на полезные Интернет-ресурсы.

Пособие можно использовать для самостоятельного изучения разделов дисциплин «Мультимедиа технологии и компьютерная графика» и «Основы веб-технологий», при выполнении практических работ, в проектной деятельности, а также при подготовке к экзаменам. Учебное пособие соответствует ФГОС ВО по направлению подготовки 09.03.02 «Информационные системы и технологии» (квалификация (степень) «бакалавр»).

Учебное пособие предназначено для студентов, обучающихся по направлению 09.03.02 «Информационные системы и технологии», всех форм обучения.

Текстовое электронное издание
Минимальные системные требования:
Компьютер: процессор AMD, Intel от 1ГГц, 100 Мб HDD, ОЗУ от 1 Гб,
Видеоадаптер от 1024 Мб, Сетевой адаптер 10/100/1000 Мб/с;
Клавиатура; Мышь; Монитор с разрешением от 800x600;
Операционная система: Windows XP SP3/Vista/7/8/10;
Программное обеспечение: Adobe Acrobat Reader, браузер Internet Explorer, Mozilla Firefox и др. Скорость подключения от 10 Мб/с.

Оглавление

Введение

Под «веб-технологиями» обычно имеют в виду разработку веб-сайтов и веб-приложений с помощью языков разметки, языков программирования, баз данных, систем управления контентом сайта и других технологий.

Веб-технологии в современном мире решают широкий круг задач, поэтому и сам процесс разработки очень многогранен. Помимо fullstack-разработчика, который полностью создает все веб-приложение, выделяют отдельные профессии frontend-разработчика, backend-разработчика, веб-верстальщик, веб-дизайнер и т.д.

В данном пособии представлен теоретический материал и практические задания, касающиеся основ верстки сайтов. Основные технологии, которыми должен владеть веб-верстальщик – это HTML5 и CSS3, также он должен разбираться в основах UI/UX-проектирования, адаптивной и отзывчивой верстки, кроссбраузерности и кроссплатформенности.

Учебное пособие «Веб-технологии: основы верстки сайтов» предназначено для бакалавров направления подготовки 09.03.02 «Информационные системы и технологии». Данное пособие может быть использовано при изучении отдельных тем дисциплины «Мультимедиа технологии и компьютерная графика», а также на начальных этапах изучения дисциплины «Основы веб-технологий».

Изучение раздела, связанного с основами верстки сайтов, направлено на формирование следующих компетенций:

– способность использовать современные информационные технологии и программные средства при решении задач профессиональной деятельности;

– способность выполнять работы по созданию (модификации) и сопровождению информационных систем.

В контексте изучения представленного раздела овладение данными компетенциями означает овладение следующими способами деятельности:

– проектирование структуры веб-страниц и структуры сайта в целом;

– создание и модификация веб-сайтов с помощью технологий верстки: языка разметки гипертекста и каскадных таблиц стилей;

– заполнение и оформление контента сайтов (текст, изображения, видео и т.д.);

– проектирование и создание модульной сетки сайта,

– создание адаптивных, доступных и кроссбраузерных сайтов.

Учебное пособие включает в себя:

– теоретический материал разделов «Введение в веб-технологии. основы HTML и CSS» и «Верста сайтов»;

– методические рекомендации по выполнению практических работ;

– перечень вопросов для самопроверки;

– перечень вопросов для углубленного изучения темы;

– тематика проектов для самостоятельного выполнения;

–ссылки на полезные Интернет-ресурсы.

Использование пособия позволяет организовать самостоятельную работу обучающихся по овладению теоретическим и практическим материалам дисциплины.

Глава 1 Введение в веб-технологии. Основы HTML и CSS

1.1 Основные понятия веб-технологий

Интернет – это всемирная информационная компьютерная сеть, связывающая между собой как пользователей компьютерных сетей, так и пользователей индивидуальных компьютеров для обмена информацией [3].

На основе Интернета работает Всемирная информационная сеть (World Wide Web, WWW) и множество других систем обмена данными (электронная почта, мессенджеры, IP-телефония, FTP-сервера и т.д.).

Одной из основных составляющих WWW являются веб-сайты. **Веб-сайт** – это одна или несколько веб-страниц, объединенных общим оформлением, контентом и имеющих уникальный адрес в сети Интернет.

Все веб-узлы в сети Интернет имеют уникальный IP-адрес. **IP-адрес** – это уникальный числовой идентификатор устройства в компьютерной сети. Каждый адрес состоит из 4 цифр (от 0 до 255), разделенных точками. Часть IP-адреса отвечает за адрес подсети, другая часть – за адрес компьютера внутри этой подсети.

IP-адрес не удобен для использования людьми, поэтому каждый веб-сайт имеет уникальное доменное имя. **Доменное имя** – это символьное имя, которое является адресом веб-сайта в Интернете. Доменные имена используются в URLs, чтобы идентифицировать сервер, на котором находится определённая веб-страница. Имя домена состоит из иерархической последовательности имён (меток), разделённых точками и заканчивающейся расширением верхнего уровня [2]. Каждая из таких меток называется доменом. Общее пространство имён Интернета функционирует благодаря Domain Name System (DNS) – системе доменных имён.

Веб-страница – это страница, созданная с помощью технологии гипертекста, языка гипертекстовой разметки HTML (Hyper Text Markup Language). Кроме HTML для разработки современных сайтов используется множество других технологий. Перечислим ключевые технологии веб-разработки:

1. Языки гипертекстовой разметки (HTML – Hyper Text Markup Language).
2. Каскадные таблицы стилей (CSS – Cascading Style Sheets).
3. Языки разработки клиентских веб-приложений (JavaScript, библиотеки JQuery, фреймворки Vue, React и др.).

4. Языки разработки серверных веб-приложений (PHP, Perl, ASP.Net и др.).

5. Технологии обмена данными (XML – Extensible Markup Language, JSON – JavaScript Object Notation).

6. Технологии хранения данных (системы управления базами данных MySQL, MS SQL Server и др.).

Большинство веб-сайтов имеют клиент-серверную архитектуру, при которой часть функционала выполняется на стороне клиента, а часть на стороне сервера. Рассмотрим структуру клиент-серверного веб-приложения (рис. 1). На стороне клиента (персональных компьютеров и мобильных устройствах) браузер формирует веб-страницу из HTML и CSS кода, полученного с сервера. Также браузер интерпретирует JS-скрипты, которые, как правило, отвечают за интерактивность пользовательского интерфейса и предобработку данных. На стороне сервера данные хранятся в базах данных. Веб-сервер выполняет всю бизнес-логику веб-приложения.

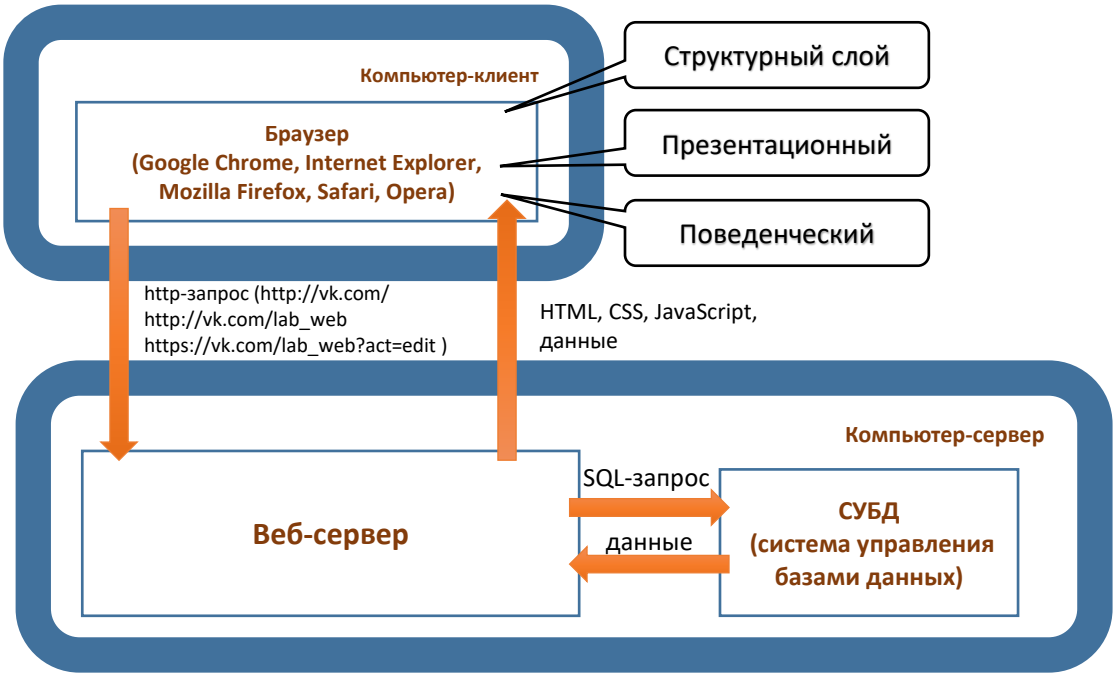


Рисунок 1 – Структура клиент-серверного приложения

Рассмотрим типовую схему работы веб-сайта. Переходя на страницу сайта или выполняя некое действие на странице (нажатие кнопки, отправка формы и другие действия) отправляется http-запрос на веб-сервер, адрес которого определяется с помощью DNS-системы. В http-запросе указывается какая страница отвечает за обработку этого запроса, а также необходимые данные для его выполнения. Веб-сервер выполняет некие алгоритмы

обработки данных, при необходимости запрашивает данные с помощью SQL-запроса из базы данных через систему управления базами данных (СУБД) на сервере данных. СУБД возвращает запрошенные данные или модифицирует (добавляет, меняет, удаляет) данные в базе данных. Затем веб-сервер отправляет данные клиенту в виде HTML-кода. Браузер преобразует HTML-код в картинку в браузере. При необходимости браузер дополнительно запрашивает у веб-сервера CSS-файлы, JS-файлы, изображения и другие файлы.

Компоненты веб-страницы:

– контент (текст, изображения, видео, аудио и т.п., в том числе из баз данных);

– логическая структура (HTML);

– оформление (CSS);

– поведение (языки программирования).

Этапы разработки веб-сайтов представлены в таблице 1.

Таблица 1 – Этапы разработки веб-сайтов

№	Этап	Результат
1	Формирование требований	Техническое задание
2	Разработка дизайна	Макет сайта (набор графических файлов)
3	Верстка сайта	Шаблон сайта (HTML, CSS, JS)
4	Разработка логики приложения (программирование)	Готовое приложение
5	Тестирование	Исправление ошибок, оптимизация
6	Публикация сайта в сети Интернет	Размещение сайта на хостинге, настройка хостинга
7	Продвижение сайта, SEO (Search Engine Optimization)	Увеличение видимости сайта в поисковых системах

Подходы к разработке сайтов:

1. Разработка «с нуля» – разработчик полностью пишет код верстки HTML, CSS, программы на языках JavaScript и PHP или других языков программирования, проектировать и создавать базу данных.

2. Использование конструкторов и систем управления контентом сайта (CMS – Content Management System), например, WordPress, Joomla, OpenCart, Bitrix, Netcat и т.д.

3. Использование фреймворков (это программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта), например, Symfony, Yii, Laravel, Aura и т.д.

1.2 Введение в HTML и CSS

Верстка сайта – это процесс создания HTML+CSS шаблона сайта.

Чаще всего в шаблон встраиваются и JS-скрипты (коды на языке программирования JavaScript), отвечающие за интерактивность веб-страниц. На сегодняшний день для разработки большинства шаблонов используются библиотеки и фреймворки, облегчающие верстку, например, фреймворк Bootstrap.

Frontend разработка – это разработка клиентской стороны веб-сайта.

Код выполняется на клиентской стороне, т.е. в браузере. Основной язык frontend разработки – JavaScript. С помощью JS-скриптов:

1. Создают интерактивные эффекты.
2. Организуют AJAX-запросы.
3. Обрабатывают и представляют данные с серверной части приложения.

Технологии верстки:

1. Языки гипертекстовой разметки HTML.
2. Каскадные таблицы стилей CSS.
3. Языки разработки клиентских веб-приложений JavaScript (а также их библиотеки и фреймворки).

Инструменты верстки:

1. Браузеры и инструменты разработки в браузере (открывается в браузере с помощью сочетания клавиш CTRL+SHIFT+I).
2. Редактор кода, например, бесплатные редакторы Visual Studio Code, Atom и многие другие.
3. Плагины для облегчения написания кода, например, Emmet.
4. Программная платформа Node.js, которая является средой выполнения JS-программ.

На рисунке 2 представлены инструменты разработки в браузере Google Chrome. Для верстки сайтов понадобятся вкладки Elements, в которых можно просмотреть HTML-код страницы, а также вкладка Style, в которой отображаются CSS-стили выделенного элемента страницы.

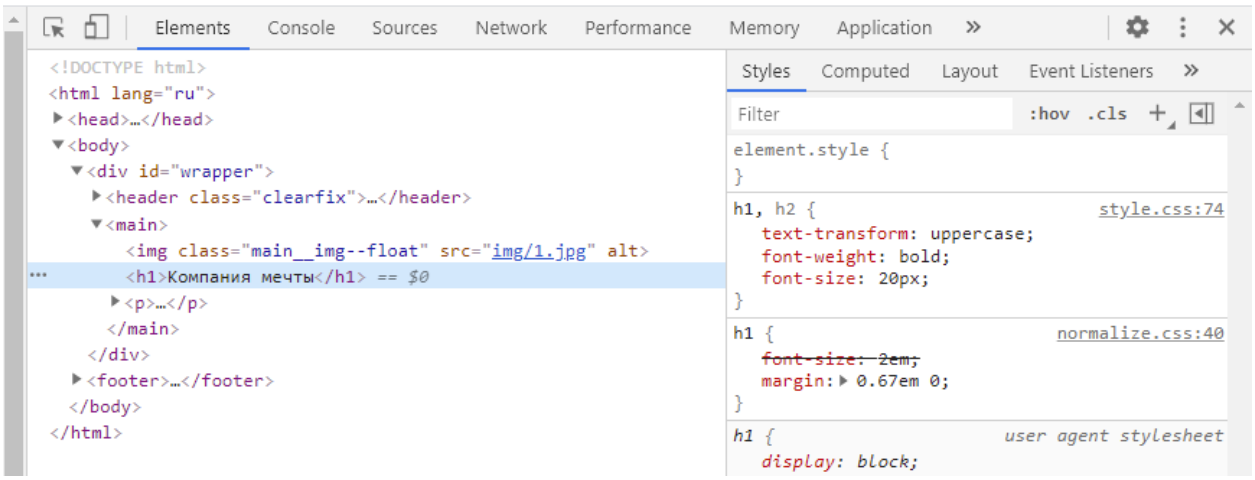


Рисунок 2 – Инструменты разработчика в браузере

Сайт – это не один файл, это множество файлов, связанных между собой. На рисунке 3 представлена типовая структура проекта сайта.

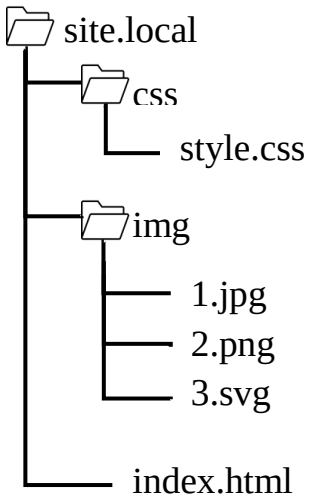


Рисунок 3 – Структура проекта

У любого сайта есть точка входа – главная страница сайта с именем index.html. CSS-стили проекта должны находиться в отдельной папке css, а картинки проекта в папке img. Имена всех файлов и папок должны содержать только строчные английские буквы, цифры, знак «_».

Структура HTML-страницы:

```
<!DOCTYPE html>
<html>
<head>
<meta charset=utf-8>
<title>Простейший документ</title>
</head>
<body text='#0000ff'  bgcolor='#f0f0f0'>
```

```
    <h1>Пример простого документа</h1>
    <hr>
</body>
</html>
```

Основной элемент языка HTML – тег. **Тег –** указатель разметки, который задается следующим образом:

```
<имя_тега>
<имя_тега атрибут1=значение атрибут2=значение …>
```

Примеры:

```
<p>
<br>
<table width="570px" align=center border=6>
<img src="img/pict.jpg">
```

Контейнер – это элемент разметки HTML:

```
<имя_тега список_атрибутов>
          содержание контейнера
</имя_тега>
```

Примеры:

```
<HTML>web-страница</HTML>
<p>абзац</p>
<table>таблица</table>
```

Есть теги, не образующие контейнер, это не закрываемые теги:

```
<img>
<hr>
<br>
```

Одни контейнеры помещаются в другие. Пример правильного вложения контейнеров:

```
<ul>
    <li>Первый элемент списка</li>
    <li> Второй элемент списка </li>
</ul>
```

Пример неправильного вложения контейнеров:

```
<ul>
    <li>Первый элемент списка</li>
    <li> Второй элемент списка
</ul></li>
```

Можно выделить следующие группы тегов:

– теги, определяющие структуру документа;

– теги оформления блоков гипертекста (параграфы, списки, таблицы, изображения);

– теги для гипертекстовых ссылок и закладок;

– теги форм для организации диалога;

– и др.

Комментарии в HTML:

```
<!- закомментированный код -->
```

Рассмотрим некоторые ключевые теги. Страница начинается с доктайпа. Доктайп задает тип документа, чтобы браузер мог определить версию HTML и правильно отобразить страницу.

Версия HTML5 (последняя актуальная версия):

```
<!DOCTYPE html>
```

Версия HTML4:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
```

Контейнер HTML-документа:

```
<html lang="ru">
</html>
```

Контейнер HEAD и его содержимое:

```
<head>
  <title>Заголовок страницы</title>
  <meta charset="название кодировки">
  <meta name="keywords" content="важные, ключевые, слова">
  <meta name="description" content="краткое описание">
  <link href="адрес_файла_стилей.css" rel="stylesheet">
</head>
```

В этом разделе устанавливают кодировку, ключевые слова, описание страницы, заголовок страницы на вкладке браузера и подключают шрифты и стили.

Контейнер BODY включает все контейнеры и элементы, которые отображаются в браузере.

Структурные элементы страницы – это блочные элементы, прежде всего div, а также семантические (смысловые) теги:

1. Main – основное содержание сайта (только один на странице).

2. Header – шапка сайта или другой части сайта (например, статьи).

3. Footer – подвал сайта или другой части сайта.

4. Nav – навигация (совокупность кнопок).

5. Section – крупный смысловой (или «логический») раздел.

6. Article – обозначает цельный, законченный и самостоятельный фрагмент информации.

7. Aside («сайдбары») – дополнительное содержание, не связанное напрямую с основным (боковые панели с разными модулями – календарь, реклама, облака тегов и т.д.).

Cascading Style Sheets (Таблицы Каскадных Стилей) – это язык, содержащий набор свойств для описания внешнего вида любых HTML документов. Стили позволяют оформить веб-страницы. Стили можно подключить несколькими способами:

1. Атрибут `style` у каждого тега.

```
<img style="width:300px;height:200px" src="img/avatar.jpg">
<p style="text-align:center;color:#fff;font-size:14px"><p>
```

2. Внутренняя таблица стилей.

Тег `style` в контейнере `head`:

```
<style type="text/css">
<!--
p {
     color: red;
}
h1 {
     font-family: Georgia, 'Times New Roman', Times, serif;
     font-size: 20px;
}
-->
</style>
```

3. Внешняя таблица стилей.

Стили создаются в отдельном файле с расширением css.

```
body {
     color: #545C66;
     background-color: #cccccc;
}
h1 {
     font-size: 1.5em;
     font-weight: normal;
}
```

Данный файл подключают к странице в разделе `head`:

```
<link href="адрес_файла_стилей" rel="stylesheet">
```

Более предпочтителен третий способ, так как одни и те же стили можно применять сразу к нескольким страницам сайта, редактировать стили можно также в одном месте.

Стилевые правила в примерах применяются к конкретным тегам, `p, h1, body` и т.д.

Комментарии в CSS:

```
/*   закомментированный код */
```

1.3 Тег div. Блочная модель документа

Существует три основных типа элементов:

1. Блочные элементы (CSS-свойство `display` равно `block`).

2. Строчные элементы (CSS-свойство `display` равно `inline`).

3. Строчно-блочные элементы (CSS-свойство `display` равно `inline-block`).

К блочным элементам относятся: `<address>`, `<article>`, `<aside>`, `<blockquote>`, `<dd>`, `<div>`, `<dl>`, `<dt>`, `<details>`, `<fieldset>`, `<figcaption>`, `<figure>`, `<footer>`, `<form>`, `<h1>-<h6>`, `<header>`, `<hr>`, `<li>`, `<legend>`, `<nav>`, `<ol>`, `<p>`, `<pre>`, `<section>`, `<table>`, `<ul>` и другие.

Блочные элементы по умолчанию занимают всю ширину родителя, перед и после себя создают разрыв строки. То есть несколько блочных элементов располагаются друг за другом по вертикали. Блочным элементам можно задать фиксированную ширину и высоту.

К строчным элементам относятся: `<a>`, `<area>`, `<b>`, `<cite>`, `<code>`, `<dfn>`, `<del>`, `<em>`, `<i>`, `<img>`, `<ins>`, `<label>`, `<map>`, `<mark>`, `<s>`, `<samp>`, `<small>`, `<span>`, `<strong>`, `<sub>`, `<sup>`, `<time>`, `<q>`, `<u>` и другие.

Строчные элементы располагаются друг за другом по горизонтали и, если не вмещаются в одну строку, переносятся на следующую. Строчные элементы имеют размер своего содержимого. Даже если явно задать у строчных элементов свойства ширины, высоты и отступов, это не повлияет на размеры и расположение элемента.

Строчно-блочные элементы: `<audio>`, `<button>`, `<canvas>`, `<input>`, `<select>`, `<textarea>`, `<video>` и другие.

Данные элементы обладают свойствами и блочных элементов (у них можно задавать высоту, ширину и отступы), и строчных (они располагаются друг за другом в ряд и при необходимости переносятся на другую строку).

С помощью свойства `display` можно менять тип любого html-контейнера. Рассмотрим пример – три блока `div`, по умолчанию у блоков `div` свойство `display` определено как `block` (рис. 4). Обратите внимание, высота блоков равна высоте их содержимого, а ширина равна ширине родителя, в данном случае всего контейнера `body`.

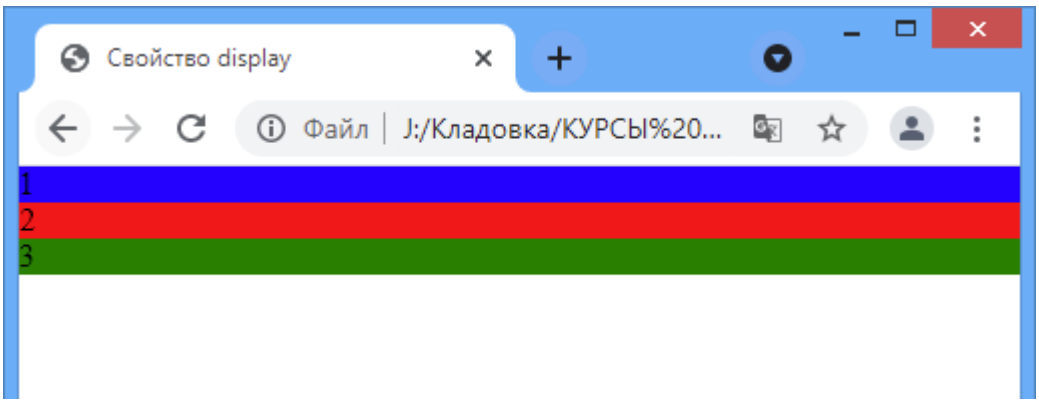
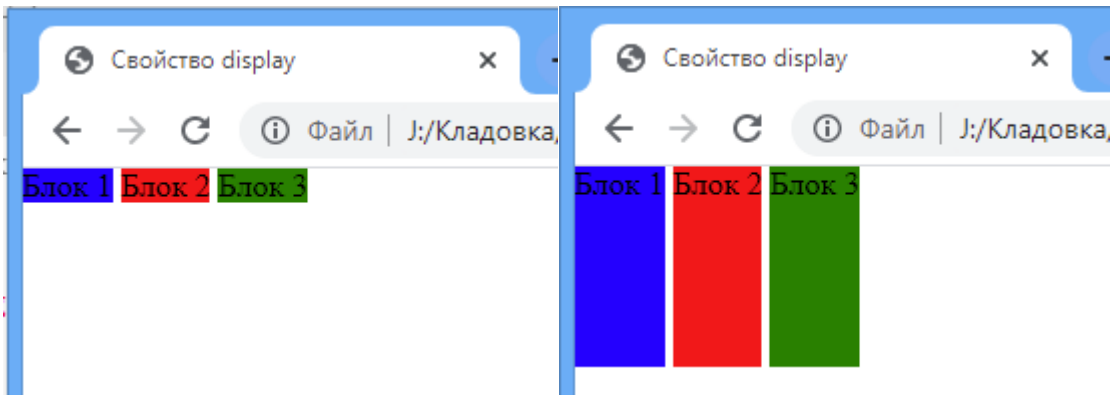


Рисунок 4 – Внешний вид блочных элементов по умолчанию

Если задать блокам явную ширину и высоту, то результат будет следующий (рис. 5) – блоки встанут в столбец друг за другом.

Рисунок 5 – Блочные элементы с заданной шириной и высотой

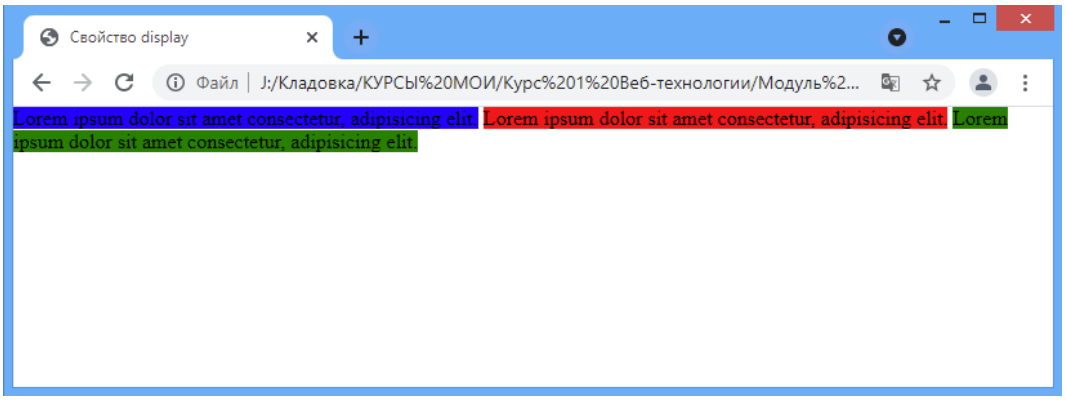
С помощью свойства `display` можно изменить тип элементов на строчные (рис. 6 а) и строчно-блочные (рис. 6 б). Обратите внимание – у строчных элементов высота и ширина равна высоте и ширине контента (несмотря на то, что высота и ширина блоков задана явно), у строчно-блочных элементов высота остается заданной, а ширина равна ширине контента.



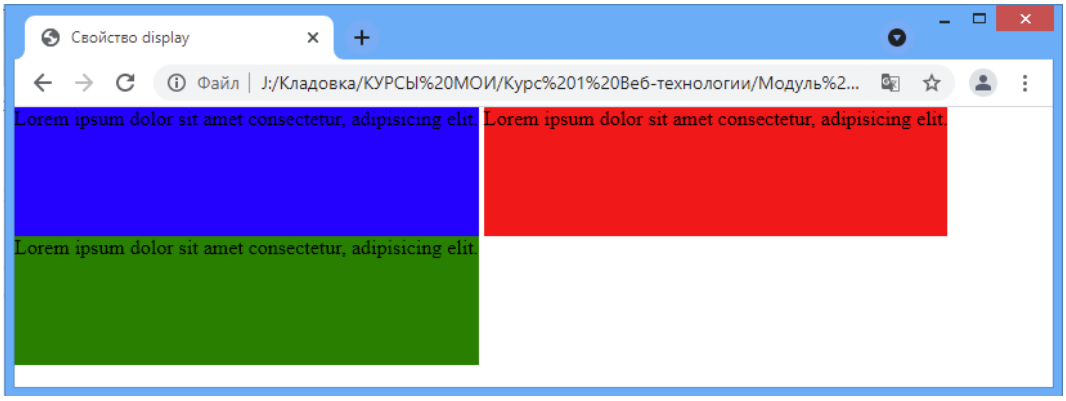
а б

Рисунок 6 – Внешний вид строчных элементов `display:inline` (а), и строчно-блочных элементов `display:inline-block` (б)

Есть еще одно принципиальное отличие строчных и строчно-блочных элементов. Увеличим контент этих блоков и посмотрим, что будет, если они не будут вмещаться в одну строку (рис. 7). Строчные ведут себя как параграфы и текст переносятся вместе с блоками, а строчно-блочные переносятся поблочно.



а



б

Рисунок 7 – Внешний вид строчных элементов `display:inline` (а), и строчно-блочных элементов `display:inline-block` (б)

Существую и другие значения свойства `display` с ними можно познакомится в документации [2].

Тег `div` – основной элемент для разработки структуры сайта. С помощью данного блока и его семантических аналогов (теги `header`, `footer`, `main`, `article` и др.) размечают как крупные блоки веб-страниц, так и реализует микроверстку отдельных блоков. Напомним, что стоит использовать тег `div` только в том случае, если никакой другой семантический тег не подходит. Все что будет далее рассмотрено в данном параграфе, касается не только тега `div`, но и прочих схожих семантических блочных тегов.

У тега `div` есть только глобальные атрибуты, т.е. атрибуты общие для всех HTML-элементов. Сам по себе тег никак не влияет на макет страницы, если у него не заданы CSS-стили.

Рассмотрим самые частые CSS-свойства для этого тега (рис. 8):

– `width` – ширина блока, по умолчанию – равна ширина родительского элемента;

– `height` – высота блока, по умолчанию – равна высоте контента;

– `padding` – внутренние отступы, по умолчанию – нулевые;

– `margin` – внешние отступы, по умолчанию – нулевые;

– `border` – рамки, по умолчанию – нулевые.

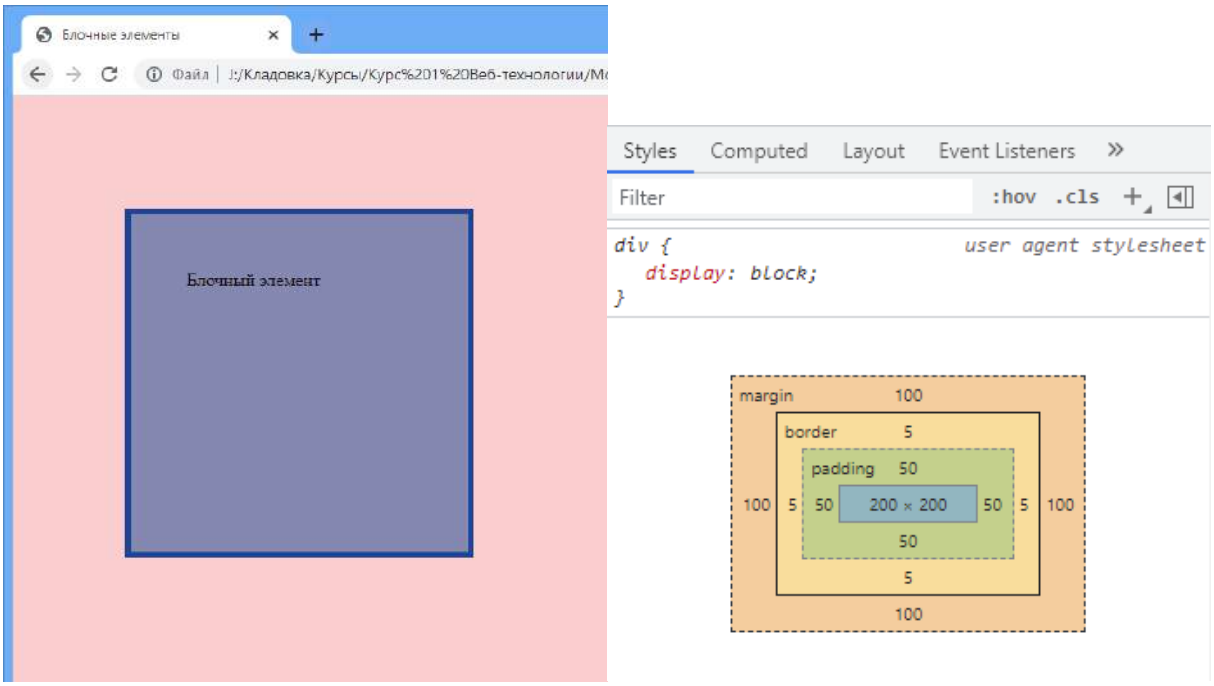


Рисунок 8 – Блочный элемент `div` и его свойства (`box-sizing: content-box`)

Согласно спецификации CSS:

`ширина блока = ширины контента (width) + значений отступов (padding) + границ (border)`

Аналогично обстоит и с высотой блока. На рисунке 8 (справа) контуры самого блока выделены сплошной черной линией, как видно они включает в себя саму ширину и высоту блока, внутренние отступы и рамку. Таким образом, наш блок имеет размеры 255 на 255 px.

Такой подход не всегда удобен, добавление рамок и отступов для оформления блоков, может привести к изменению размера блока на страницы, а это может повлечь за собой изменение расположения соседних элементов. Свойство `box-sizing` позволяет изменить этот алгоритм. Это свойство имеет два значения:

1. `content-box` – значение по умолчанию соответствует стандартной блочной модели: ширина и высота элемента равна заданным свойствам `width` и `height`, а значение рамок и внутренних отступов добавляется дополнительно.

2. `border-box` – изменяет режим расчёта ширины и высоты элемента: теперь ширина и высота элемента включает и рамку, и внутренние отступы и, собственно, ширину и высоту содержания самого элемента (рис. 9):

`ширина блока = width`

`ширины контента = width - значений отступов (padding) - границ (border)`

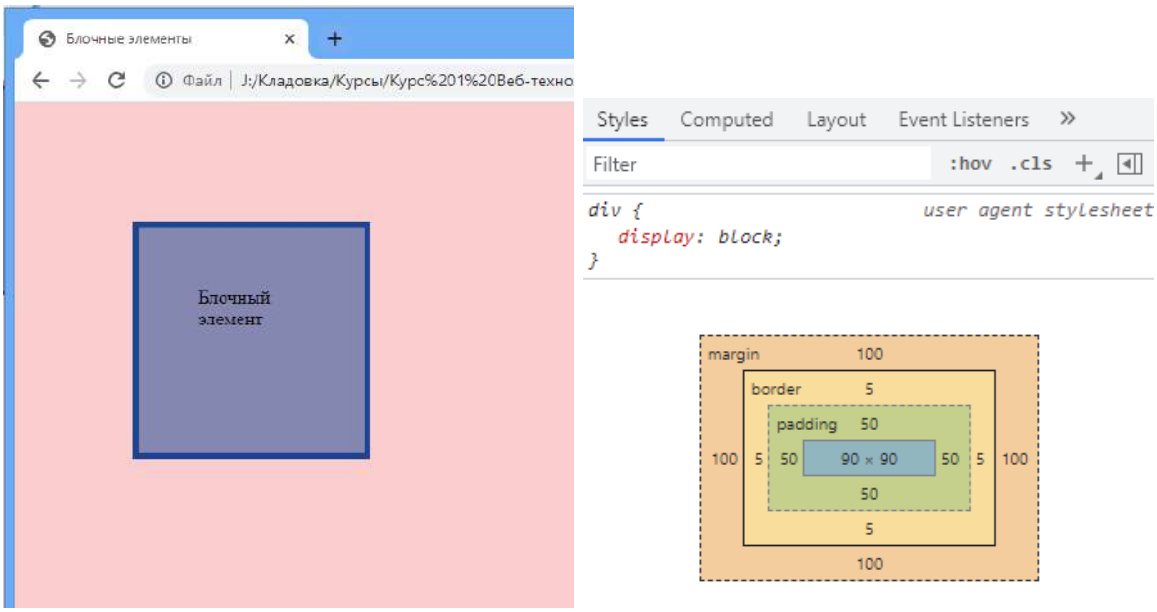


Рисунок 9 – Блочный элемент `div` и его свойства (`box-sizing: border-box`)

Как видно из рисунка 9, теперь размер блока равен 200 на 200 px, а контентная часть может занимать 90 на 90 px, так как из размера блока вычли величину внутренних отступов и ширину рамок.

Рассмотрим подробнее свойства `margin` и `padding`. Внешние отступы можно задать с помощью CSS-свойства `margin` следующими способами:

- `margin: x;` – одинаковые внешние отступы со всех сторон. Например, `margin: 10px;` или `margin:2em;`
- `margin: y x;` – отступ сверху и снизу равен `y`, а слева и справа – `x`. Например, `margin: 10px 20px;`
- `margin: y1 x y2;` – отступ сверху равен `y1`, слева и справа – `x`, снизу – `y2`. Например, `margin: 10px 20px 30px;`
- `margin: y1 x1 y2 x2;` – отступ сверху равен `y1`, справа – `x1`, снизу – `y2` слева – `x2`. Например, `margin: 10px 20px 30px 40px;`

Также можно использовать простые CSS-свойства:

- `margin-top: y;`
- `margin-right: x;`
- `margin-bottom: y;`
- `margin-left: x;`

Аналогично задаются и внутренние отступы (свойство `padding`).

Для определения рамок используется составное свойство `border`:

```
border: color width style;
```

Например:

```
border: red 2px solid;/*сплошная черная рамка толщиной 2px*/
```

Для задания рамки необходимо указать эти три параметра в любом порядке. Часть параметров можно опускать, но обязательно задавать стиль рамки. Также есть простые свойства: `border-style`, `border-width`, `border-color`, `border-top`, `border-bottom`, `border-right`, `border-left`. А также свойств для задания скруглены углов `border-radius`. С этими и другими свойствами можно подробнее познакомится в документации [3].

Рассмотрим вопрос горизонтального выравнивания блоков и их содержимого. Если элемент блочный, он легко центрируется относительно своего родителя, при соблюдении следующих правил:

1. У выравниваемого элемента должно быть задано CSS-свойство `margin: 0 auto;` (внешние отступы сверху и снизу – 0, можно задать и не нулевое значение, а слева и справа – автоматические, одинаковые).
2. У самого элемента должна быть задана ширина, CSS-свойство `width`.
3. У родителя также должна быть задана ширина, поэтому принято у контейнеров `html` и `body` задавать ширину `width:100%;`

Пример выравнивания представлен на рисунке 10. Красный блок выровнен относительно синего блока, а синий – относительно контейнера `body`.

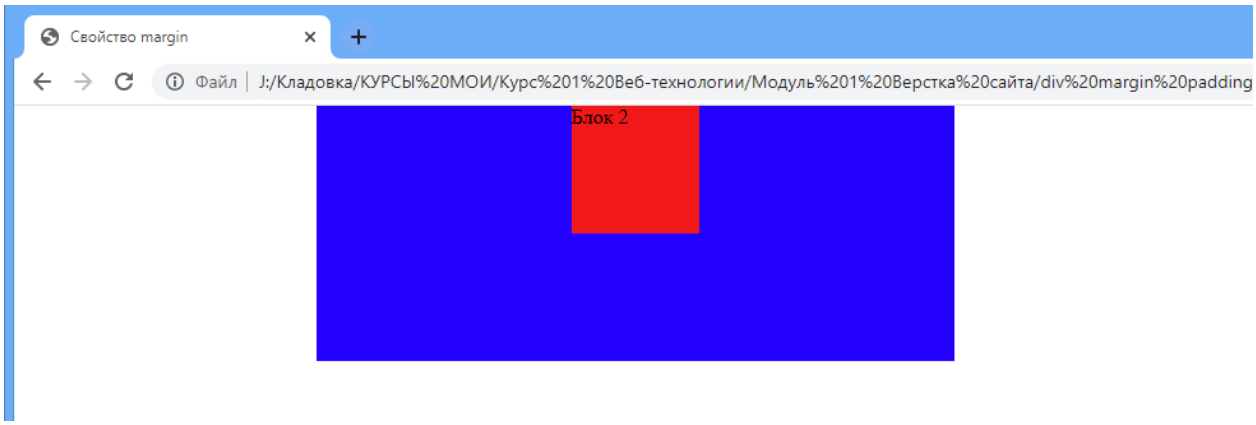


Рисунок 10 – Выравнивание с помощью `margin:0 auto;`

Если нужно выровнять строчный элемент внутри блочного или сами строчные элементы, то используется CSS-свойство:

```
text-aling: center | left | right | justify
```

Вертикальное выравнивание не рассматривается в этой главе. Во второй главе будут рассмотрены современные способы выравнивания как по вертикали, так и по горизонтали с помощью flexbox.

Рассмотрим некоторые особенности внешних отступов – это схлопывание и выпадение внешних отступов. Эффект схлопывания вертикальных отступов состоит в следующем: если два блока располагаются друг за другом по вертикали, у первого есть внешний отступ снизу, например, 100px, а у второго – отступ сверху, например, 50px. То их отступы не суммируются и не будут равны 150px, а будут равны наибольшему из отступов, в нашем случае 100px (рис. 11).

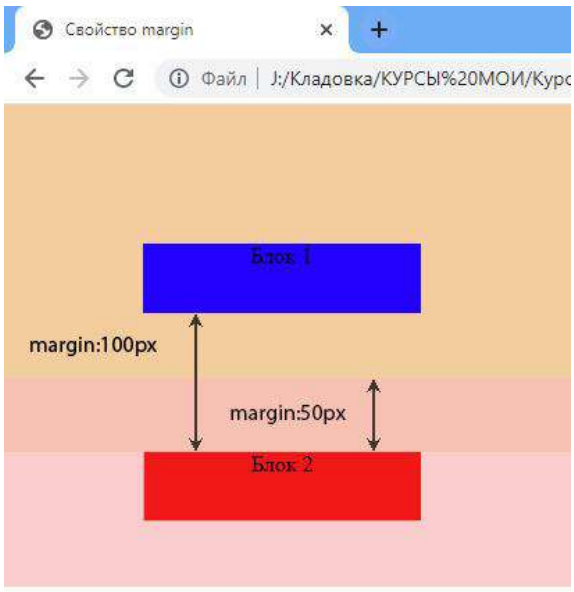


Рисунок 11 – Схлопывание вертикальных отступов

Второй эффект, выпадение внешних отступов, происходит, когда один блок расположен внутри другого и у внутреннего (вложенного) блока внешние вертикальные отступы больше чем внешний отступ родителя. В этом случаев внешний вертикальные отступы родителя становится равен отступу дочернего блока (рис. 12).

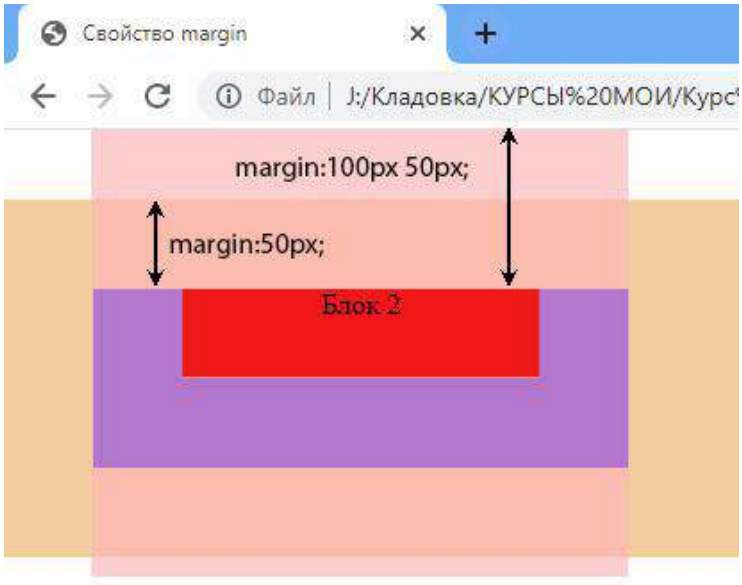


Рисунок 12 – Выпадение внешних отступов

1.4. Практическая работа 1. Введение в HTML и CSS

Цель работы: создать проект сайта, создать шаблон страниц сайта.

Задачи работы:

1. Научиться создавать структуру проекта – систему файлов и папок.
2. Изучить основные теги разметки страницы (header, footer, main, section, article, div и т.д.).
3. Научиться применять стили к селекторам тегов, идентификаторов и классов.
4. Узнать возможности редактора кода Visual Studio Code и браузера Google Chrome.

Инструменты: Visual Studio Code, браузер Google Chrome

Задание 1. Создание основы проекта

1. Создайте структуру проекта как на рисунке 13.

Рисунок 13 – Структура проекта

2. Запустите программу Visual Studio Code.
3. Выполните команду меню *Файл => Открыть папку* и выберите папку с проектом.
4. Создайте новый файл и сохраните его в корень проекта с именем index и расширением HTML.
5. Создайте новый файл и сохраните его в папку css с именем style и расширением CSS. Теперь проект имеет следующую структуру (рис. 14).

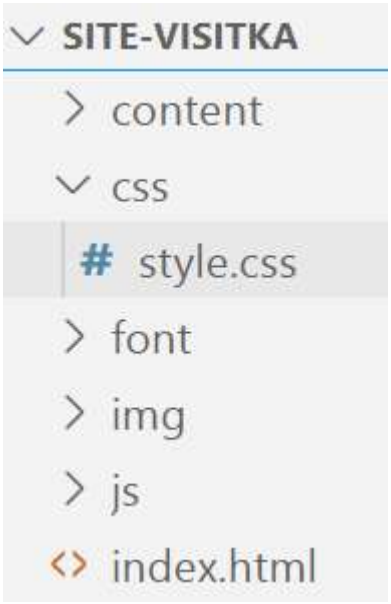


Рисунок 14 – Добавление файлов в проект

6. В документе index.html введите ! и нажмите Enter. Создастся готовая структура html-страницы (рис. 15).

7.

index.html > ...

```
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="UTF-8">
5       <meta name="viewport" content="width=device-width, initial-scale=1.0">
6       <title>Document</title>
7   </head>
8   <body>
9
10  </body>
11  </html>
```

Рисунок 15 – Структура html-страницы

8. Измените значение атрибута `lang` у тега `html` на `ru`. Сохраните изменения.

9. Подключите таблицу стилей в разделе `head`. Для этого создайте пустую строку сразу за тегом `head`, введите `link` и нажмите Enter. Укажите адрес таблицы стилей (рис. 16).

```html
<!DOCTYPE html>
<html lang="ru">
<head>
    <link rel="stylesheet" href="css/style.css">
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>

</body>
</html>
```

Рисунок 16 – Подключение файла стилей

10. Добавьте в файл style.css следующие стили:

```css
* {
    margin:0;
    padding: 0;
}
html,body {
  height: 100%;
}
header, main, footer {
    border: black solid 1px;
}
#wrapper{
    min-height: 100%;
    height: auto !important;
    height: 100%;
    width: 100%;
}
header{
    height:120px;
}
main{
    padding-bottom: 100px;
}
footer{
    margin: -100px auto 0;
    position: relative;
    height: 100px;
}
```

Задание 2. Создание шаблона главной страницы

Главная страница сайта будет выглядеть как на рисунке 17.

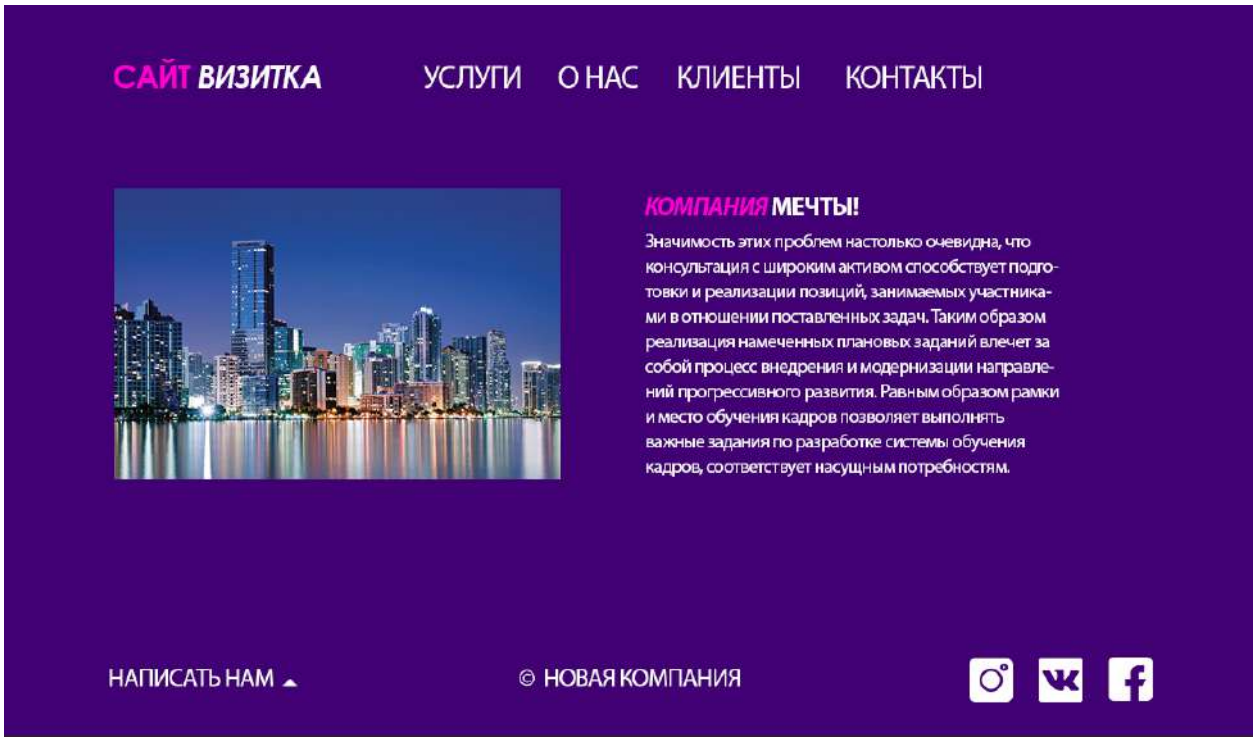


Рисунок 17 – Шаблон главной страницы сайта

У данного сайта четко выражена структура, состоящая из трех блоков – шапка, основная часть и подвал.

1. С помощью тегов `header`, `main` и `footer` создайте структуру сайта как на странице.

2. Для того, чтобы прижать подвал к низу страницы, оберните контейнеры `header` и `main` в тег `div` с `id = "wrapper"` (рис. 18).

```
<body>
    <div id="wrapper">
        <header>
            Сайт визитка
        </header>
        <main>
            Контент
        </main>
    </div>
    <footer>
        &copy; Новая компания
    </footer>
</body>
```

Сайт визитка

Контент

© Новая компания

Рисунок 18 – HTML-код и результат в браузере

Далее будет наполнять только основную часть сайта – контейнер main.

3. В папку img проекта скопируйте все изображения для будущего сайта.

4. Добавьте в контейнер main изображение 1.jpg с помощью тега `img`.

Примечание: тег `img` не является контейнером, т.е. у него нет закрывающего тега.

5. Разместите заголовок первого уровня с текстом «Компания мечты!».

6. С помощью кнопки расширения  установите плагин Рыбный текст.

7. После заголовка напишите `fishtag` и нажмите Enter, после этого выберите тег `p` и нажмите ОК, создастся параграф с тестовым текстом (рис. 19).

```
<main>
    <img src="img/1.jpg" alt="">
    <h1>Компания мечты</h1>
    <p>Значимость этих проблем настолько очевидна, что курс на
    социально-ориентированный национальный проект требует анализа поэтапного и
    последовательного развития общества. Значимость этих проблем настолько
    очевидна, что рамки и место обучения кадров требует анализа экономической
    целесообразности принимаемых изменений. Значимость этих проблем настолько
    очевидна, что консультация с широким активом проверки влечёт за собой
    интересный процесс внедрения модернизации направлений прогрессивного
    развития.</p>
</main>
```

Рисунок 19 – Наполнение основной части главной страницы

8. Просмотрите результат в браузере, убедитесь, что картинка, заголовок и параграф отображаются корректно.

Задание 3. Оформление страницы

1. Для тега `body` установите стилевые правила для фона страницы и цвета текста. Можно установить свои цвета. Сохраните, просмотрите результат.

```
body{
    background-color: #380577;
    color: white;
}
```

2. Для селекторов `header`, `main` и `footer` задайте ширину `width:1040px;`. Эти селекторы уже определены в таблице стилей, нужно только добавить нужные стили.

3. Для селектора `header, main` и `footer` задайте внешние отступы `margin:0 auto;`. Это позволит центрировать эти блок на странице.

4. Удалите все стили для рамок блоков – `border`.

5. Назначьте у картинки атрибут `class= "main__img—float"`.

6. Создайте стиль для этого класса

```
.main__img--float{

}
```

7. Для данного селектора задайте ширину – 550px.

8. Для того же селектора задайте внешние отступы справа – 40px.

9. Также добавьте стилевое правило `float:left;`. Что делает это правило?

10. Просмотрите результат работы в браузере (рис. 20).

11. Откройте инструменты разработчика в браузере, попробуйте изменить цвет фона и текста на странице с помощью вкладки Styles.

Рисунок 20 – Результат первых трех заданий

Задание 4. Создание страницы услуг

Итоговый вид страницы услуг представлен на рисунке 21.

Рисунок 21 – Шаблон страницы услуг

1. Сделайте копию страницы index.html и переименуйте ее uslugi.html.
2. Очистите контейнер `main`. Сам контейнер оставьте.
3. Между контейнерами `header` и `main` создайте блок `div` с `id="panorama"`.
4. Задайте стилевые правила для данного идентификатора:

```
#panorama {
    width: 100%;
    height: 150px;
    background: url(../img/2.jpg) no-repeat center 70%;
    background-size: cover; /*масштабирование фона*/
}
```

5. В контейнере `main` создайте контейнер `section` с `class="services"`.
6. В этой секции создайте контейнер `article` с классом `class="services__item"`.
7. Разместите картинки 3.jpg, 4.jpg, 5.jpg, 6.jpg в папку content. Так как эти изображения относятся к содержанию сайта, а не к самому шаблону.
8. В контейнере `article` разместите картинку 3.jpg из папки контент, заголовок 2 уровня с названием услуги, параграф текста и параграф с текстом «Подробнее…».
9. Создайте еще два контейнера `article` с таким же классом. Наполните их материалами (см. рисунок 21).

10. У картинок задайте класс `services__img` и задайте для этого класса ширину картинок 240px.

11. Для класса `services__item` задайте стилевые правила:

– внешние отступы слева и справа по 40px;

– ширина блока – 240px;

– `float:left;`

12. Для класса `services` задайте стили:

– внутренние отступы по 35px;

– рамку в 5px;

13. Наблюдается выпадение блоков из контейнера из-за использования `float`. Решим проблему через создание класса псевдораспорки:

```
.clearfix::after {
    content: "";
    display: table;
    clear: both;
}
```

14. Добавьте класс `clearfix` контейнеру `section`.

15. Убедитесь, что блоки больше не выпадают из контейнера и удалите рамку у блока `section`.

16. Создайте внизу контейнера main заголовок 1 уровня с текстом «Услуги компании».

17. Задайте класс у этого заголовка `class = "h1–invisible"`.

18. Установите стили для этого класса `visibility: hidden;`.

19. Просмотрите результат работы (рис. 22).

Рисунок 22 – Результат четвертого задания

Задание 5. Создание страницы «О нас»

Оформите только основную часть этой страницы, примерно, как на рисунке 23. Шапку и подвал оставьте как в предыдущих заданиях.

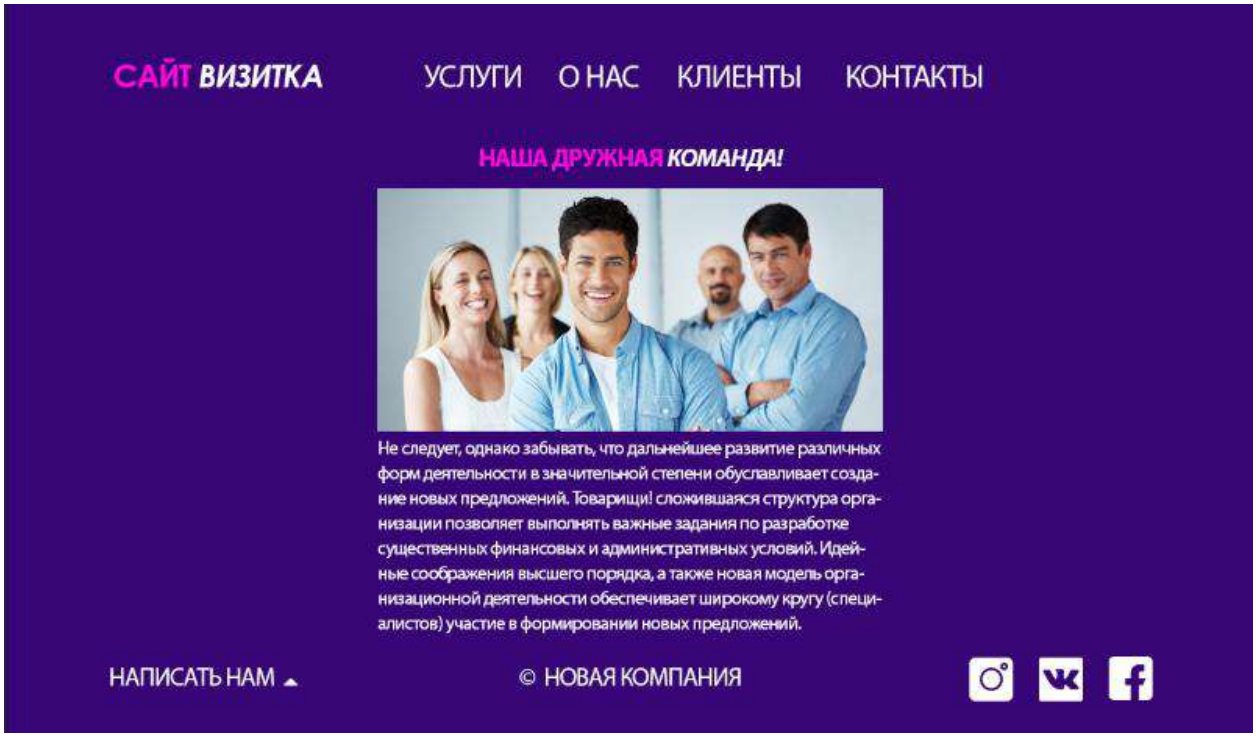


Рисунок 23 – Шаблон страницы «О нас»

Задание 6. Создайте страницу «Контакты»

Оформите только основную часть этой страницы, примерно, как на рисунке 24. Шапку и подвал оставьте как в предыдущих заданиях.

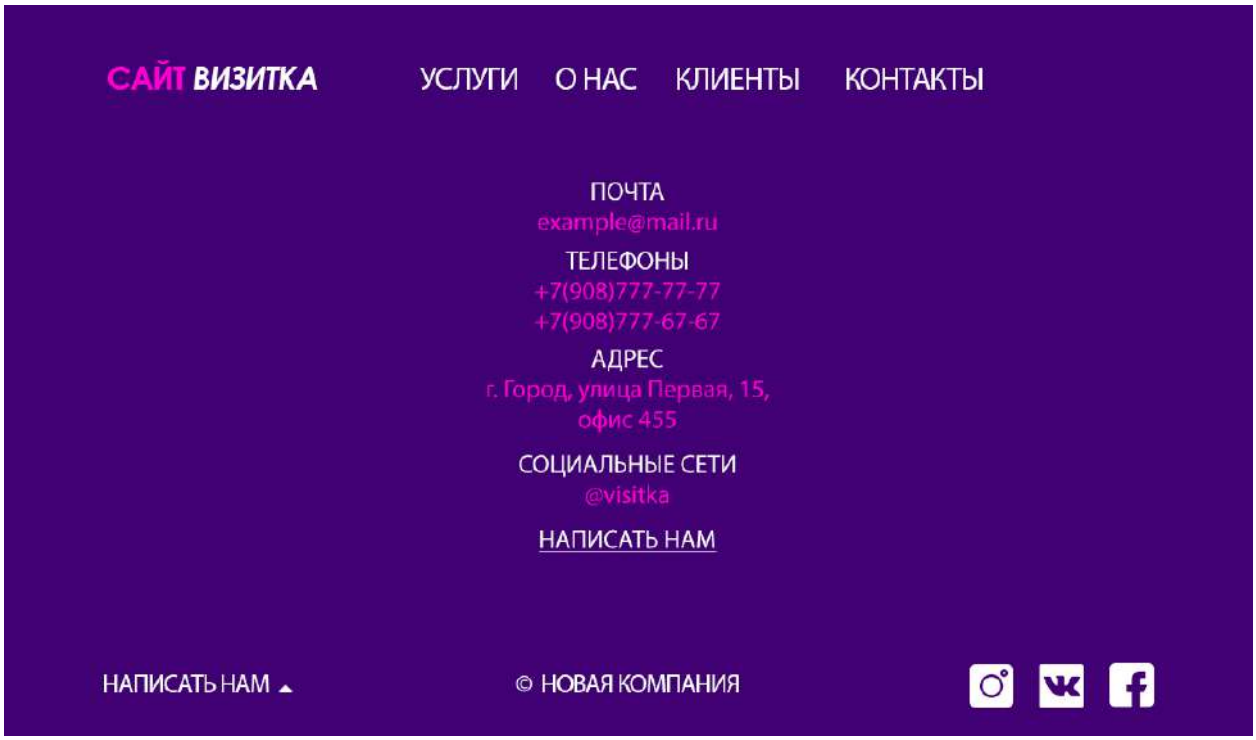


Рисунок 24 – Шаблон страницы контактов

Задание 7. Создание страницы «Клиенты»

Самостоятельно придумайте шаблон контентной части для страницы «Клиенты». Оформите только основную часть этой страницы. Шапку и подвал оставьте как в предыдущих заданиях.

1.5. Оформление контента сайта

К контенту сайта относятся:

- текст,
- ссылки,
- изображения,
- таблицы,
- видео,
- аудио,
- анимация.

Теги, для оформления текста:

1. Параграф `<p>`

2. Заголовки разных уровней `<h1>,…, <h6>`
3. Текстовый блок `<span>`
4. Теги оформления текста:

```
<p>Тэг-выделитель для <mark>подсветка</mark> текста</p>
<p><del>Удаленный текст</del></p>
<p><s>Зачеркнутый</s></p>
<p><ins>Строка - дополнение к документу</ins></p>
<p><u>Подчеркнутая</u></p>
<p><strong>Жирный текст</strong></p>
<p><b>Жирный текст</b></p>
<p><em>Курсив</em></p>
<p><i>Курсив</i></p>
```

5. Подстрочные и надстрочные символы

```
<p>x<sup>2</sup></p>
```
x^2

```
<p>H<sub>2</sub>O</p>
```
H_2O

6. Цитаты в тексте

```
<q>цитата в тексте</q>
```

7. Блочная цитата

```
<blockquote>
   Блок с цитатой
   <cite>источник</cite>
</blockquote>
```

8. Тег задания даты и времени

```
<time datetime="2020-08-19T10:00">
10:00 19.08.2020
</time>
```

9. Тег задания адреса

```
<address>улица Пушкина, дом Колотушкина </address>
```

CSS-правила для оформления текста:

1. Размер текста `font-size:16px;`
2. Высота линии текста (межстрочный интервал) `line-height: normal;`
3. Гарнитура (тип) шрифта `font-family: "PT Sans", "Arial", sans-serif;`
4. Толщина шрифта `font-weight: bold;`
5. Горизонтальное выравнивание текста `text-align: center; /* left, right, justify */`
6. Вертикальное выравнивание текста `vertical-align: middle; /* top, bottom, baseline*/`
7. Цвет текста `color: black;`
8. Оформление текста `text-decoration: underline;`

9. Стиль текста `font-style:italic;`

10. Трансформация текста `text-transform: uppercase;`

Единицы измерения, используемые в верстке:

1. Абсолютные (px, cm, mm и др.):

```
font-size: 1cm;
font-size: 38px;
width:500px;
```

2. Относительные (em, rem, % и др.):

```
font-size: 2em;
font-size: 2rem;
width:50%;
```

1em – это размер текущего шрифта. 1 rem – размер шрифта, установленного у элемента `<html>`.

Способы задания цвета:

1. Табличные цвета – это цвета, имеющие названия. Посмотреть эти цвета можно на сайте colorscheme.ru.

2. Шестнадцатеричный код – #RRGGBB или #RGB: #e0b743, #ffbb00 (#fb0).

3. Функция rgb(r, g, b) – rgb(255, 0, 0).

4. Функция rgba(r, g, b, a) – rgba(0, 0, 0, 0.5).

Типы шрифтов:

1. Serif – шрифты с засечками (антиквенные), типичный пример –Times New Roman;

2. Sans-serif – рубленные шрифты (шрифты без засечек или гротески), типичный пример – Arial;

3. Cursive – курсивные шрифты;

4. Fantasy – декоративные шрифты;

5. Monospace – моноширинные шрифты, ширина каждого символа в таком семействе одинакова (например, шрифт Courier).

Оформление списков в HTML:

1. Неупорядоченные списки – теги `ul` и `li`. Пример:

```
<p>Горячие напитки:</p>
<ul>
     <li>чай</li>
     <li>какао</li>
     <li>кофе</li>
</ul>
```

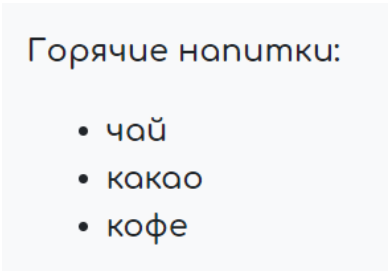


Рисунок 25 – Пример неупорядоченного списка

2. Упорядоченные списки – теги `ol` и `li`. Пример:

```
<p>Рейтинг моих любимых горячих напитков:</p>
<ol>
        <li>кофе</li>
        <li>чай</li>
        <li>какао</li>
</ol>
```

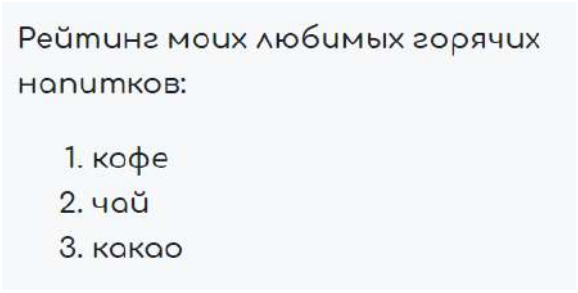


Рисунок 26 – Пример упорядоченного списка

3. Списки описания – теги `dl, dt` и `dd`. Пример:

```
<dl>
     <dt>Термин 1</dt>
     <dd>Определение термина 1</dd>
     <dt>Термин 2</dt>
     <dd>Определение термина 2</dd>
</dl>
```

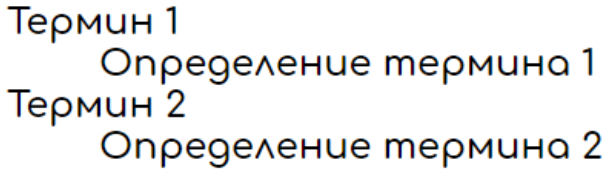


Рисунок 27 – Пример списка определений

Списки можно вкладывать одни в другие. Пример:

```
<ol>
   <li>1
```

```
        <ul>
            <li>1.1</li>
            <li>1.2</li>
        </ul>
    </li>
    <li>2</li>
</ol>
```

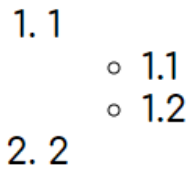


Рисунок 28 – Пример вложенных списков

Ссылка (гиперссылка) – это элемент страницы, ссылающаяся на элемент в этом документе или на другой документ внутри этого сайта или вне его.

Ссылки задаются с помощью тега a:

```
<a href="URL">...</a>
```

Примеры:

```
<a href="img/xxx.jpg">Посмотрите на эту фотографию!</a>
<a href="article1.html">Статья 1</a>
```

Адреса, которые можно задавать ссылкам:

1. Абсолютные

```
<a href="C:/site/article1.html">Статья 1</a>
<a href="http://htmlbook.ru/">Справочник по HTML</a>
```

2. Относительные

```
<a href="article1.html">Статья 1</a>
```

Виды ссылок:

1. Якоря – это ссылки на элементы внутри этой же страницы.

```
<img src="logo.png" id="top">
…
<a href="#top">наверх</a>
```

2. Внутренние ссылки – ссылки на ресурсы данного сайта (картинки, файлы, другие веб-страницы).

```
<a href="index.html">Главная</a>
```

3. Внешние ссылки – ссылки на другие ресурсы в сети Интернет.

```
<a href="http://htmlbook.ru/">Справочник по HTML</a>
```

Изображения на страницы задаются с помощью тега img:

```
<img src="logo.png">
<img src="logo.png" width="300px">
```

```
<img src="logo.png" width="300px" height="300px">
<img src= "photo.png" alt="Фотография автора">
<a href= "about.html">
     <img src="photo.png" alt="Фотография автора">
</a>
```

Картинка с описанием:

```
<figure>
  Схема, график, диаграмма или код
  <figcaption>Подпись к содержимому</figcaption>
</figure>
```

Форматы графических файлов, используемых на веб-страницах:

1. Растровые форматы:
   a. JPG для полноцветных изображений без прозрачности;
   b. GIF для изображений с палитрой до 256 цветов, содержащие прозрачность и анимацию;
   c. PNG для полноцветных изображений с градиентной прозрачностью;
2. Векторный формат:
   a. SVG.

1.6. Практическая работа 2. Работа с контентом сайта

Цель работы: Оформить содержимое сайта с помощью тегов и стилевых правил.

Задачи работы:

1. Изучить основные теги для задания текстовых блоков, изображений и ссылок.
2. Изучить некоторые стилевые правила оформления текста, картинок и ссылок.

Инструменты: Visual Studio Code, браузер Google Chrome.

Задание 1. Оформление логотипа сайта

Логотип сайта, расположенный в шапке сайта, всегда является ссылкой на главную страницу сайта (index.html)

1. Откройте в редакторе кода главную страницу сайта – index.html.
2. Оформите текст «Сайт визитка» в шапке сайта как ссылку на главную страницу:

```
<header>
    <a href="index.html">Сайт визитка</a>
</header>
```

3. Задайте для этого тега a идентификатор `id= "logo"`.

4. Для данного идентификатора в файле style.css задайте стиль трансформации текста:

```
#logo{
    text-transform: uppercase;
}
```

5. Также для данного селектора задайте следующие стилевые правила:

– убрать подчеркивание: `text-decoration:none;`

– цвет текста белый: `color:#fff;`

– полужирный шрифт: `font-weight: bold;`

– высота линии текста: `line-height: 100px;`

– размер шрифта: `font-size: 24px;`

6. С помощью тега `i` выделите слово «визитка» курсивом.

7. Слово «сайт» поместите в тег span и создайте у него класс `class = "pink"`.

8. Для класса `.pink` задайте цвет текста – розовый.

Задание 2. Создание меню сайта

Обычно навигацию создают с помощью неупорядоченных списков.

1. Создайте неупорядоченный список:
   – услуги,
   – о нас,
   – клиенты,
   – контакты.

2. Каждый элемент списка сделайте ссылкой на соответствующую страницу.

```
<ul>
    <li><a href="uslugi.html">Услуги</a></li>
    ...
</ul>
```

Сейчас список должен выглядеть как на рисунке 29. Данный список будет оформлен в практической работе №3.

Рисунок 29 – Внешний вид списка для главного меню

3. Продублируйте содержание тега header на все страницы

Задание 3. Оформление подвала

1. Очистите подвал.
2. Создайте в подвале два параграфа:

– первый параграф с текстом: Написать нам и символом ▲

– второй параграф с текстом: «© Новая компания».

3. Добавьте в подвал блок `div` после двух параграфов.
4. В блок поместите три картинки – иконки социальных сетей (рис. 30). Иконки социальных сетей необходимо предварительно разместить в папке img.

Рисунок 30 – Результат добавление блоков в подвал

5. У параграфов задайте классы `footer__block`.
6. У блока div задайте класс `footer__icons`.
7. У иконок задайте классы `footer__img`.
8. Для классов `footer__block` и footer__icons задайте стилевые правила:

– тип отображения блочно-строчное, чтобы блоки выстроились в ряд: `display: inline-block;`

– ширина блоков 335px.

9. Для класса footer__icons задайте стилевое правило выравнивания текста по правому краю: `text-align: right;`
10. Для класса `footer__img` задайте стилевое правило ширины в 50px.
11. Для классов `footer__block` добавьте стилевое правило для оформления текста: сделать все буквы прописными.
12. Для иконок сделать внешние отступы слева по 10px.

13. Сделайте иконки гиперссылками на страницы социальных сетей. У ссылок задайте атрибут `target= "_blank"` (страница будет открываться на отдельной вкладке).

14. Продублируйте содержание подвала на все страницы (рис. 31).

Рисунок 31 – Результат выполнения задания

Задание 4. Оформление контентной части

1. Просмотрите в практической работе 1 внешний вид страниц сайта и с помощью тега `span` и класса `pink` сделайте, где это необходимо, текст розовым.

2. Для заголовков первого и второго уровня установите стилевые правила:

– трансформацию – все заглавные;
– полужирный шрифт;
– размер шрифта – 20px.

3. На страницы «Услуги» параграфы Подробнее… сделайте ссылками. Адреса ссылок пока оставьте пустыми. Пока не оформляйте эти ссылки.

4. В тексте на главной странице, на странице «О нас» и на странице «Услуги» выделите отдельные фразы с помощью тегов: `strong, em, mark` и других при необходимости.

5. Задайте для тега `mark` стилевое правило, которое будет задавать цвет подсветки текста, которая будет подходить под дизайн сайта.

6. Оформите картинки на сайте с помощью стилевых правил:

– `border,`
– `border-radius,`
– `box-shadow.`

Описание данных правил можно найти в HTML и CSS справочниках [1, 2, 4, 5].

1.7.Основы CSS

Основной элемент каскадных таблиц стилей – это **селектор**. Селектор определяет к каким HTML-элементам применить CSS-правило. Основные типы селекторов:

1. Селекторы по тегам;
2. Селекторы по классам;
3. Селекторы по идентификаторам.

Таблица 2 – Примеры селекторов

Селектор	CSS-код	HTML-код

Селекторы по тегам	тег {
	стилевые правила
	}
		<тег></тег>

Селекторы по классам	.имя_класса{
	стилевые правила
	}
		<тег class="имя_класса" >
		</тег>

Селекторы по идентификаторам	#имя_идентификатора{
	стилевые правила
	}
		<тег id= "имя_идентификатора">
		</тег>

Какого цвета будет каждый параграф в примере ниже?

Таблица 3 – Примеры селекторов

CSS-код	HTML-код

p{
color: red;
}
.blue{
color: blue;
}
#last{
color: green;
}
	Текст первого параграфа
	
	Текст второго параграфа
	
	Текст третьего параграфа
	
	Текст четвертого параграфа

Первый параграф будет – красного цвета. Второй параграф – синего цвета, так как приоритет селекторов классов выше чем у селекторов по тегам. Третий блок текста также будет синий. Четвертый параграф будет зеленого цвета, так как приоритет селекторов идентификаторов выше всех остальных.

Структура стилевых правил:

```
селектор{
     стилевое правило 1: значение 1;
     стилевое правило 2: значение 2;
     …
}
```

Стилевые правила могут быть:

1. Простые

```
color:red;
background-image: url(img/bg.jpg);
font-size: 2em;
```

2. Составные

```
background: url(images/hand.png) repeat-y #fc0;
border:  black solid 2px;
padding: 20px 30px 50px 40px;
```

Основные свойства CSS:

1. Наследование – перенос правил форматирования для элементов, находящихся внутри других. Существуют наследуемые свойства (например, размер и тип текста), которые передаются дочерним элементам от родителей. И не наследуемые (например, рамки, отступы). Какие свойства стилей наследуются, а какие нет, можно посмотреть в спецификации [2].

2. Каскадирование – одновременное применение разных стилевых правил к элементам документа – с помощью подключения нескольких стилевых файлов, наследования свойств и других методов.

К одному элементу можно применить сразу несколько классов. Это называется мультиклассом (пример в таблице 4). Текст параграфа будет красного цвета и увеличенного размера в 24px.

Таблица 4 – Пример мультиклассов

HTML-код	CSS-код

`<p class= "red big">`
`  Текст`
`</p>`
	`.red {`
	`  color:red;`
	`}`
	`.big {`
	`  font-size:24px;`
	`}`

Другие типы селекторов:

1. **Универсальный селектор** – стили применяются ко всем элементам на странице.

```
* {
     margin:0;
```

```
        padding:0;
    }
```

2. **Селекторы потомков** применяются ко всем элементам данного типа, находящимся внутри родительского элемента, независимо от глубины вложенности.

Таблица 5 – Селекторы потомков

CSS-код	HTML-код

nav a{
 font-size: 1.5em;
}
#main_menu h2{
 font-weight: normal;
}
.content img{
 width:500px;
 float: left;
}
	
	 <nav id="main_menu">
	  Заголовок
	
	   <a…>Пункт меню 1
	   <a…>Пункт меню 1
	
	 <img class="logo" src="img/pict.jpg">
	 </nav>

Стилевые правила селектора `nav a` будут применены ко всем ссылкам внутри тега `nav`. Стилевые правила селектора `#main_menu h2` применятся к заголовкам второго уровня внутри тега с `id="main_menu"`. Стилевые правила селектора `.content img` применятся к картинкам внутри тега с классом `class="content"`.

3. **Дочерние селекторы** применяются ко всем элементам данного типа, которые являются дочерними непосредственно по отношению к указанному элементу.

Таблица 6 – Дочерние селекторы

CSS-код	HTML-код

nav a{
 font-size: 1.5em;
}
nav>a{
 font-weight: bold;
}
nav li>a{
 color: red;
}
	
	 <nav id="main_menu">
	 <a…>Название<a>
	
	   <a…>Пункт меню 1
	   <a…>Пункт меню 1
	
	 </nav>

Стилевое правило для селектора потомков `nav a` будет применено ко всем трем ссылкам, а стилевое правило для дочерних селекторов `nav>a` будет применено только к ссылке с текстом «Название», так как ссылки пунктов меню не являются дочерними по отношению к контейнеру `nav`. Для них выполнится правило для селектора `nav li >a`.

4. **Соседние селекторы** позволяют выбирать элементы, которые находятся на этом же уровне вложенности после указанного элемента, с тем же родителем.

Таблица 7 – Соседние селекторы

CSS-код	HTML-код

b+i {
 color:red;
}
p+.about{
 border: red solid 3px;
}
	<img class="about" src="…">
	
	 Простотекст
	
	 Просто текст
	
	<img class="about" src="…">

В стилевом правиле для соседних селекторов `b+i` стиль применится для тега `i` в первом параграфе, т.к. в нем до тега `i` идет тег `b`. А стилевое правило `p+.about` применится только для второй картинки с классом `about`, так как перед ней находится параграф.

5. **Селекторы атрибута** выбирают все элементы, имеющие данный атрибут или атрибут с определённым значением.

Таблица 8 – Селекторы атрибутов

CSS-код	HTML-код

a[href="index.html"]{
 color:red;
}
img[src$="png"]{
 width:100px;
}
img[src^="content/"]{
 botder: 1px solid red;
}
	
	 <img src="content/1.png">
	
	
	 <img src="img/1.jpg">
	

Селектор `a[href="index.html"]` выбирает все теги `a`, у которых атрибут `href` равен `index.html`. Селектор `img[src$="png"]` выбирает все

теги `img`, у которых атрибут `src` заканчивается png, то есть выбрать все картинки с расширением png. Селектор `img[src^="content/"]` выбирает все теги `img`, у которых атрибут `src` начинается с `content`, то есть выбрать все картинки, загруженные из папки `content`. Есть и другие правила задания селекторов атрибутов.

6. **Псевдо классы** определяют динамическое состояние элементов, которое изменяется с помощью действий пользователя, а также его положением в дереве документа.

```
селектор: псевдокласс {Описание правил стиля}
```

Примеры псевдо классов:

:active (происходит при активации пользователем элемента);

:link (применяется к не посещённым ссылкам, т.е. таким ссылкам, на которые пользователь ещё не нажимал);

:hover (активизируется, когда курсор мыши находится в пределах элемента, но щелчка по нему не происходит);

:visited (применяется к посещённым ссылкам);

:first-child (применяется к первому дочернему элементу селектора, который расположен в дереве элементов документа);

:lang(язык) (определяет язык, который используется в документе или его фрагменте).

Примеры:

```
a { //убираем у ссылок подчеркивание
 text-decoration: none;
}
a:hover { //при наведении на ссылку у нее появляется подчеркивание
 text-decoration: underline;
}
li:first-child{ // стиль задается первому элементу списка
  background-color: red;
}
p:lang(en){//стиль задается параграфам со атрибутом lang="en"
    font-size:16pt;
}
```

7. **Псевдо элементы** позволяют задать стиль элементов не определённых в дереве элементов документа, а также генерировать содержимое, которого нет в исходном коде текста.

```
селектор:псевдоэлемент { Описание правил стиля }
```

Примеры псевдоэлементов:

:after (применяется для вставки назначенного контента после содержимого элемента);

:before (применяется для вставки назначенного контента до содержимого элемента);

:first-letter (определяет стиль первого символа в тексте элемента, к которому добавляется);

:first-line (определяет стиль первой строки блочного текста).

Примеры:

```
//после параграфа с классом new добавляется текст -New
p.new:after {
    content: "- New!"; }
//задается стиль для первой строки параграфа
p:first-line {
    font-family: Arial, Helvetica, sans-serif;
    font-size: 90%;
    color: black; }
//задается стиль для первой буквы параграфа
p:first-letter {
    font-family: "Times New Roman", Times, serif;
    font-size: 200%;
    color: red;}
```

Кроме описания стилей селекторов в CSS-файлах можно задавать директивы (@-правила). Директива – это конструкция, которая позволяет создавать в CSS инструкции для изменения отображения либо поведения элементов страницы.

```
@директива правило
```

Некоторые директивы:

@charset - Применяется для задания кодировки внешнего CSS-файла. Это имеет значение в том случае, если в CSS-файле используются символы национального алфавита.

@import - Позволяет импортировать содержимое CSS-файла в текущую стилевую таблицу.

@media - Указывает тип носителя, для которого будет применяться указанный стиль. В качестве типов выступают различные устройства, например, принтер, коммуникатор, монитор и др.

Примеры:

```
@charset "utf-8"; //установка кодировки
@import url("mystyle.css") //подгрузка стилей
@media screen { //медиа-запрос для полноэкранного режима
     .module { width: 50%;}
}
@media print { //медиа-запрос для режима печати
     .module { width: 60%; }
```

```
}
//медиа-запрос для экранов до 480px
@media max-device-width: 480px{
     .module { width: 100%; }
}
```

У стилевых правил есть приоритеты. Приоритеты стилей в порядке возрастания приоритета:

1. Стиль браузера.
2. Стиль автора.
3. Стиль пользователя.
4. Стиль автора с добавлением !important.
5. Стиль пользователя с добавлением !important.

У селекторов тоже есть приоритеты, они определяются специфичностью селектора. Если к одному элементу одновременно применяются противоречивые стилевые правила, то более высокий приоритет имеет правило, у которого значение специфичности селектора больше.

Специфичность – это некоторая условная величина, вычисляемая следующим образом:

1. за каждый идентификатор начисляется 100 условных единиц;
2. за каждый класс и псевдо класс начисляется 10 условных единиц;
3. за каждый селектор тега и псевдо элемент начисляется 1 условная единица.

Складывая указанные значения, получим значение специфичности для данного селектора.

Примеры:

```
* {} 0+0+0=0 специфичность=0
p {} 0+0+1=1 специфичность=1
p>b {} 0+0+2=2 специфичность=2
ul li.active {} 0+10+2=12 специфичность=12
.nav .item {} 0+20+0=20 специфичность=20
#logo {} 100+0+0=0 специфичность=100
```

1.8. Практическая работа 3. Основы CSS

Цель работы: изучить особенности создания каскадных таблиц стилей.

Задачи работы:

1. Понять принципы каскадирования и наследования стилевых правил.
2. Научиться создавать стили для селекторов потомков, псевдо элементов, псевдо классов.
3. Научиться рассчитывать специфичность стилей.

4. Научиться подключать шрифты на страницы.

Инструменты: Visual Studio Code, браузер Google Chrome.

Задание 1. Шрифтовое оформление страницы

1. Найдите шрифт на сервисе Google Fonts (https://fonts.google.com/) для основного текста на странице.

2. В разделе language выберите Cyrillic (шрифты, с поддержкой русских букв) и выберите нужные категории шрифтов.

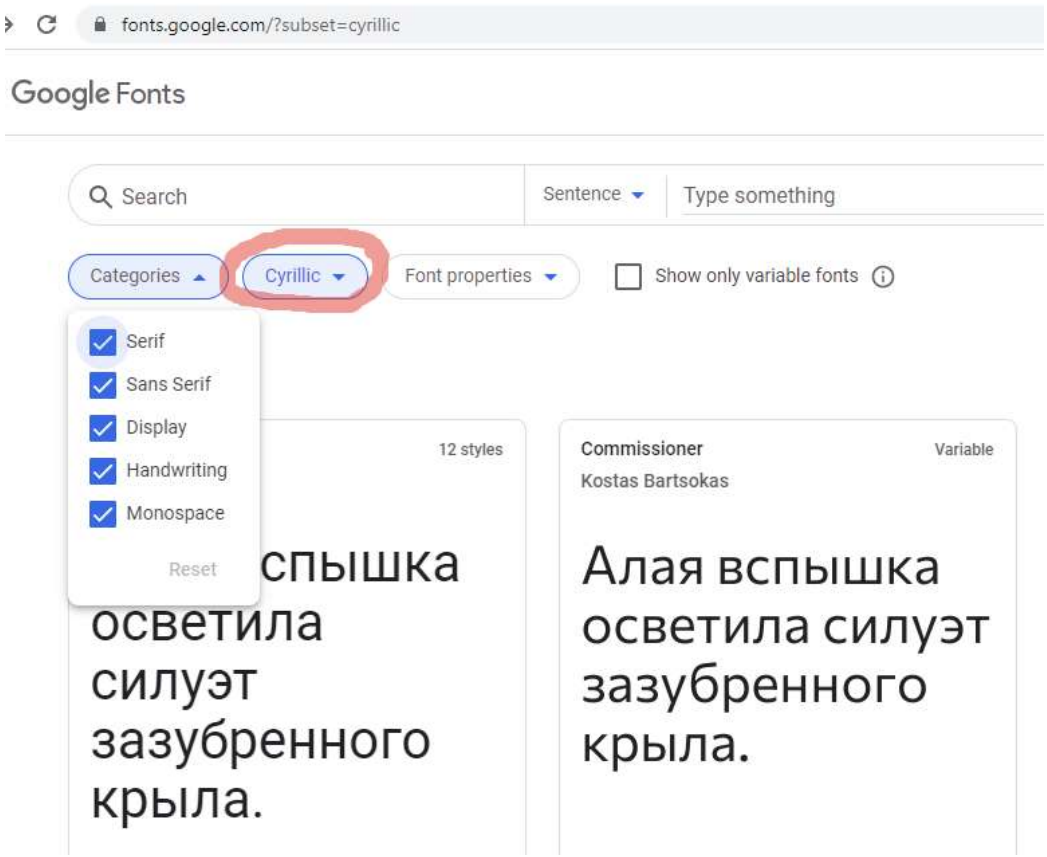


Рисунок 32 – Выбор шрифта – шаг 1

3. Выберите понравившийся шрифт.

4. Если панель Selected family закрыта, то нажимаем кнопку справа вверху Show your selected families.

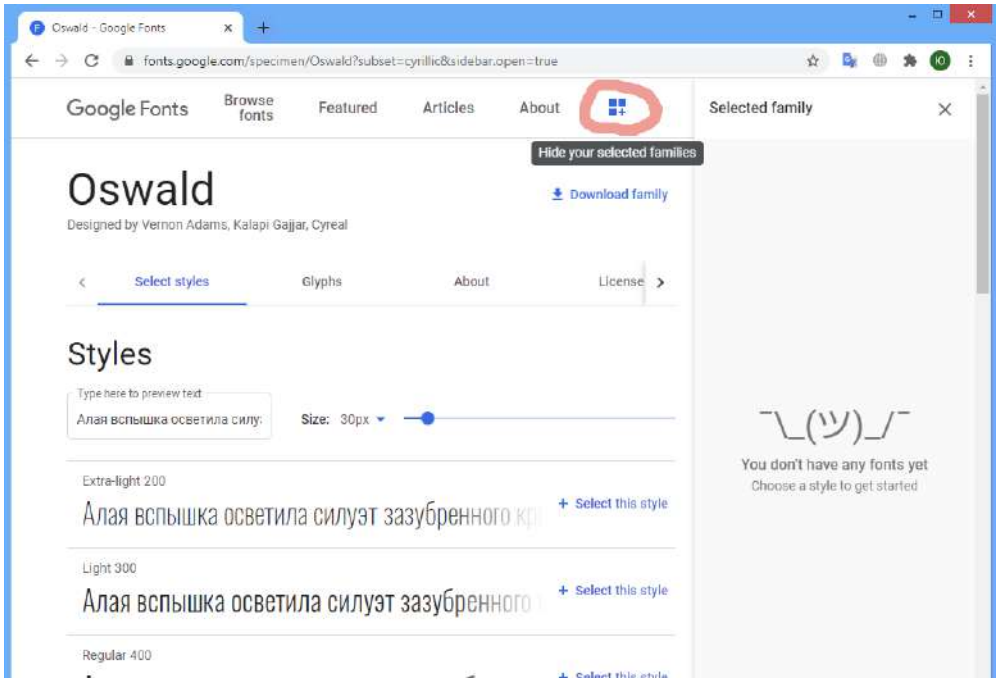


Рисунок 33 – Выбор шрифта – шаг 2

5. Выберите нужные шрифты с помощью кнопки Select this style.

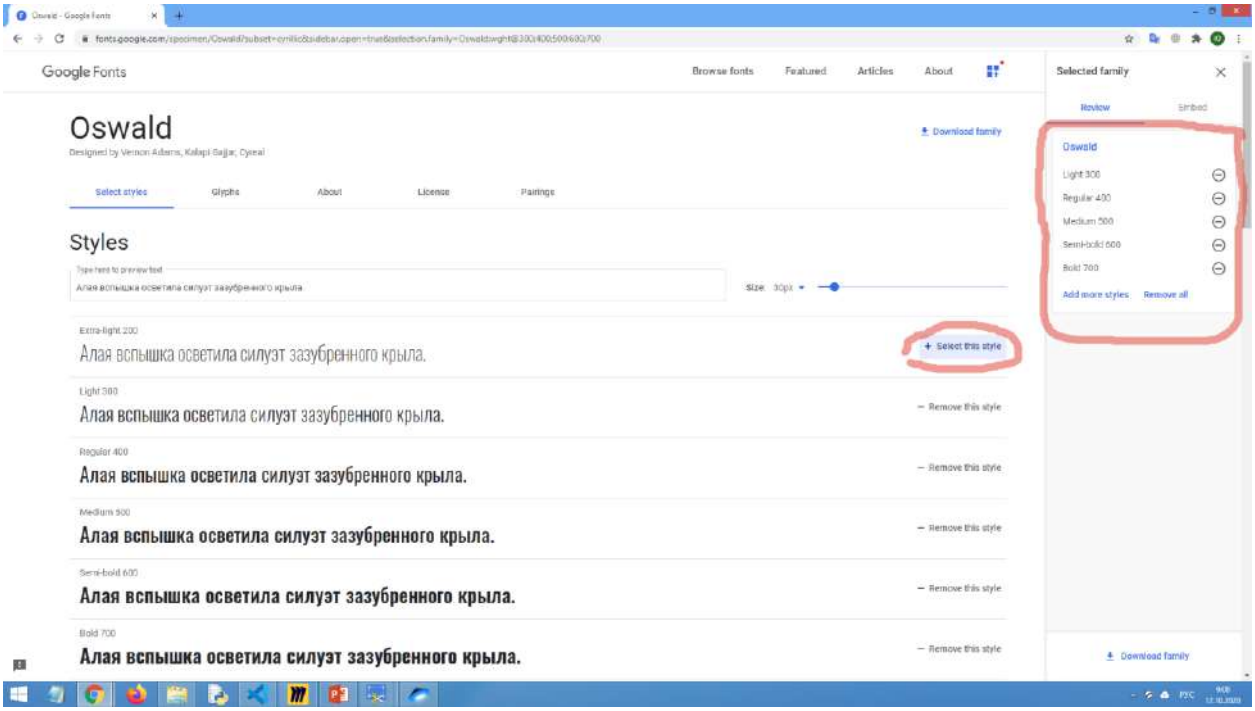


Рисунок 34 – Выбор шрифта – шаг 3

6. Перейдите на вкладку Embed скопируйте ссылку для подключения шрифтов.

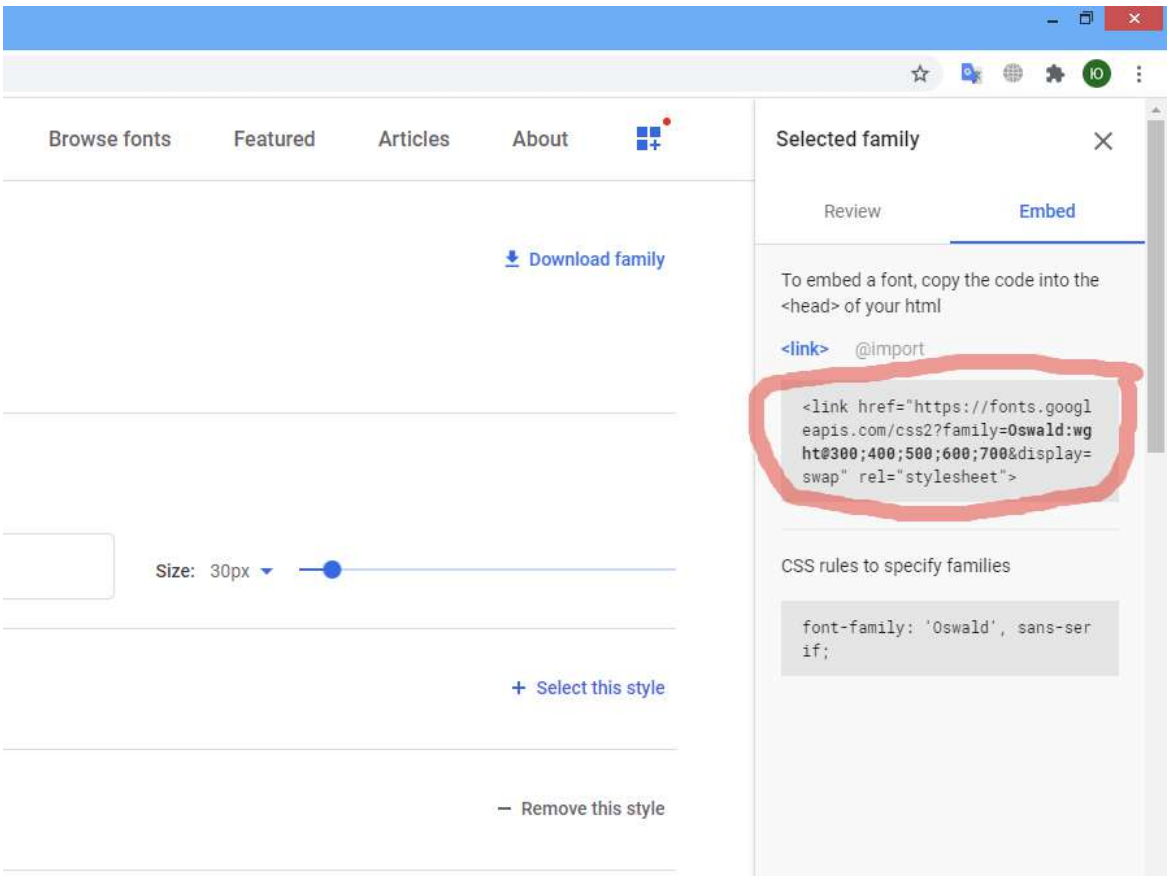


Рисунок 35 – Выбор шрифта – шаг 4

7. Подключите шрифт на страницу до подключения ваших стилей.
8. В стилевых правилах для тега `body` задайте шрифт для основного содержимого страницы. Как задать CSS-правило также представлено на вкладке Embed. Например, если вы подключили шрифт Oswald, то стиль для `body` будет выглядеть так:

```
font-family: 'Oswald', sans-serif;
```

Совет: Кроме подгруженного шрифта обязательно задавайте стандартные шрифты, на случай если браузер не сможет подгрузить внешние шрифты. Ниже представлены некоторые стандартные семейства шрифтов:

```
.serif{
      font-family: Times, 'Times New Roman', Georgia, serif;
}
.sans-serif{
      font-family: Verdana, Arial, Helvetica, sans-serif;
}

.monospace{
      font-family: 'Lucida Console', Courier, monospace;
}
```

Если вы подгрузили шрифт без засечек, то стоит указать следующее стилевое правило:

```
font-family: 'Oswald', 'Verdana', 'Arial', 'Helvetica', sans-serif;
```

9. Исправьте стилевые правила для `body`. `font-family` является наследуемым правилом и будет применено ко всем текстовым элементам на странице.

10. Найдите шрифт для заголовков, установите его для заголовков всех уровней. Таким образом вы переопределите шрифтовое оформление для заголовков.

Задание 2. Оформление основного меню

1. Измените html-код для главного меню

```
<nav id="mainnav">
    <ul>
        <li><a href="uslugi.html">Услуги</a></li>
        <li><a href="aboutus.html">О нас</a></li>
        <li><a href="klienti.html">Клиенты</a></li>
        <li><a href="kontakti.html">Контакты</a></li>
    </ul>
</nav>
```

Так как на странице может быть не одно меню, стоит у каждого задавать свой идентификатор. Стилевые правила для элементов меню будем задавать с помощью правил для селекторов потомков.

2. Для идентификатора `logo` задайте правило `float:left`, а для тега `header` задайте класс `clearfix`.

3. Теперь представим меню в виде горизонтального списка с помощью селекторов потомков:

```
#mainnav ul {
    list-style: none; /*убираем маркеры списка*/
    margin: 0;
    padding-left: 0;
}
#mainnav li{
    display: inline; /*выстраиваем пункты меню в ряд*/
}
#mainnav a {
    text-decoration: none; /*убираем подчеркивание*/
}
```

4. Оформите меню с помощью отступов и стилей для оформления текста, чтобы оно выглядело как на рисунке 36.

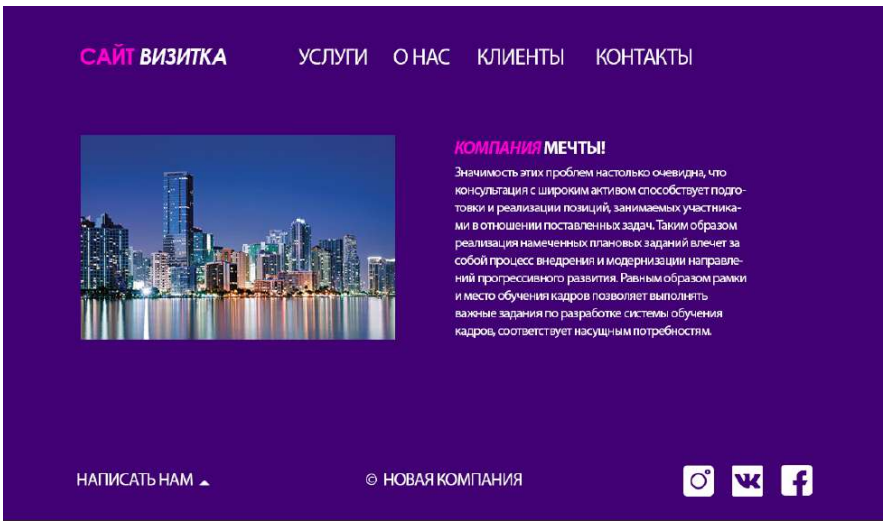


Рисунок 36 – Главная страница сайта

5. С помощью псевдо класса `:hover` добавьте подчеркивание к ссылкам меню при наведении курсора.

```
#mainnav a:hover{
    text-decoration: underline;
}
```

6. Чтобы выделять текущий пункт меню (ту страницу на которой вы находитесь) создайте стилевые правила для ссылки внутри элемента `li` с классом `active`, которые находятся внутри главного меню:

```
#mainnav li.active a{
    font-weight: bold;
}
```

7. Продублируйте измененное меню на все страницы. На каждой странице для активного (текущего) пункта `li` задайте класс `active`.

Задание 3. Использование различных селекторов

1. С помощью псевдо класса второго дочернего элемента у селекторов класса `footer__block` задайте стилевое правило – выравнивание текста по центру.

2. Только на странице «Услуги» в разделе услуг (класс `servises`) с помощью селекторов потомков оформите ссылки Подробнее… розовым цветом без подчеркивания.

3. Для тех же ссылок создайте правило, чтобы при наведении подчеркивание появлялось.

4. Для заголовка третьей услуги задайте класс `new`. Для этого класса создайте псевдо элемент `after`:

```css
.new::after{
    content:"new";
    font-size: 10px;
    vertical-align: super;
    background-color: violet;
    border-radius: 2px;
    padding: 3px;
    margin-left: 5px;
}
```

Подумайте за что отвечает каждое CSS-правило.

5. Измените стилевые правила:

– выделите стилевые правила одинаковые для всех секций на всех страницах и задайте для них отдельный класс или селектор тега `section;`

– специфичные стилевые правила выделите в отдельные классы, например, `services, contacts` и т.д.

– аналогично поступите с другими повторяющимися тегами, например, `article`.

6. Проверьте оформление всех страниц, оно должно соответствовать рисунку 37.

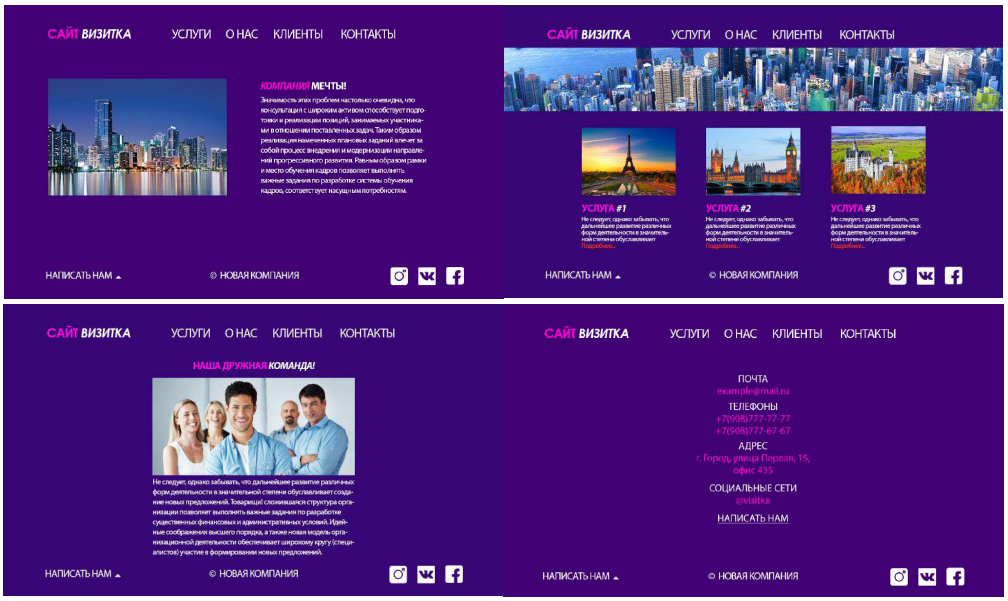


Рисунок 37 – Страницы сайта

7. Рассчитайте специфичность следующих правил:

– `body`
– `.pink`
– `#logo`
– `#mainnav ul`
– `#mainnav .active`

- #mainnav a:hover
- #mainnav ul li.active

1.9. Таблицы и формы

Структура **таблиц** в HTML:

```
<table border=1>
  <thead>
    <tr> <th>Пункт 1</th><th>Пункт 2</th></tr>
  </thead>
  <tbody>
    <tr><td>1</td><td>2</td></tr>
    <tr><td>3</td><td>4</td></tr>
  </tbody>
</table>
```

Вся таблица помещается внутрь контейнера `table`, который включает в себя контейнеры заголовка таблицы `thead` (не обязательный контейнер) и тело таблицы `tbody`. Заголовок и тело таблицы состоит из строк `tr`, а те в свою очередь из ячеек заголовка `th` (для заголовка таблицы) или ячеек таблицы td (для тела таблицы).

Чтобы создать более сложную структуру таблицы нужно объединять строки и столбцы таблицы.

Объединение столбцов (рис. 38):

```
<table border=1>
  <thead>
    <tr> <th>Пункт 1</th><th>Пункт 2</th></tr>
  </thead>
  <tbody>
    <tr><td colspan=2>12</td></tr>
    <tr><td>3</td><td>4</td></tr>
  </tbody>
</table>
```

Пункт 1	Пункт 2
12	
3	4

Рисунок 38 – Объединение столбцов

Объединение строк (рис.39):

```
<table border=1>
 <thead>
```

```
    <tr><th>Пункт 1</th><th>Пункт 2</th></tr>
  </thead>
  <tbody>
    <tr><td>1</td>
        <td rowspan=2>24</td></tr>
    <tr><td>3</td></tr>
  </tbody>
</table>
```

Пункт 1	Пункт 2
1	24
3	

Рисунок 39 – Объединение строк

Заголовок таблицы задается с помощью тега `caption` (рис. 40):

```
<table>
  <caption>Заголовок</caption>
  <tr>
    <td>...</td>
  </tr>
</table>
```

Заголовок

Пункт 1	Пункт 2
1	2
3	4

Рисунок 40 – Заголовок таблицы

Для удаления псевдотрехмерного эффекта у рамок таблицы их необходимо схлапывать с помощью стилевого правила `border-collapse`. Пример задания границ таблицам (рис. 41)

```
table {
    border-collapse: collapse;
    border: 5px solid gray;
}

td, th {
    border: 2px solid black;
}
```

Пункт 1	Пункт 2
1	2
3	4

Рисунок 41 – Границы таблицы

Для оформления таблиц очень часто используются псевдоклассы:

`:first-child` – первый дочерний элемент,

`:last-child` – последний дочерний элемент,

`:nth-child(odd)` – все нечетные дочерние элементы,

`:nth-child(even)` – все четные дочерние элементы,

`:nth-child(3n)` – каждый третий дочерний элементы,

`:nth-child(3n+2)` – каждый третий дочерний элементы начиная со второго,

`:nth-child(3)` – третий дочерний элемент.

Например, с помощью псевдо классов можно реализовать через строчное оформление таблицы (рис. 42).

```
tr:nth-child(even){
  background-color:#8642da;
}
td:first-child{
  width: 40%;
}
td:nth-child(3){
  text-align: left;
}
```

Пункт 1	Пункт 2	Пункт 3	Пункт 4
1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Рисунок 42 – Чересстрочное оформление таблицы

Веб-форма является специально ограниченной областью на странице сайта. В эти области посетитель сайта может внести ту или иную информацию, а также выбрать конкретные действия из предложенных.

```
<form id="form1" action="обработчик">
…
</form>
```

В примере ниже данные формы отправляются на обработку на сервер, алгоритм обработки данных находится в файле action.php на сервере.

```
<form id="form1" action="action.php" method="POST">
   …
   <input type="submit" value="Отправить" >
</form>
```

В примере ниже данные формы отправляются в JS-скрипт, код обработки находится в функции btnClick(). В дальнейшем сам скрипт может их отправить на сервер.

```
<form id="form1">
   …
   <input type="button" value="Отправить"
onClick="btnClick();">
</form>
```

Примеры элементов формы представлены на рисунке 43.

Рисунок 43 – Примеры элементов формы

Основные элементы форм:

1. `form` – устанавливает форму на веб-страницу, все элементы формы должны быть внутри контейнера `form`. У форм и у всех ее элементов

необходимо задавать уникальные идентификаторы для будущей обработки форм программами на разных языках программирования.

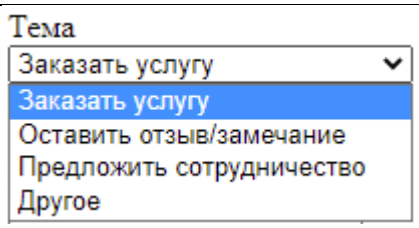
2. `label` – устанавливает подпись (метку) к элементам формы и связывает метку с элементом по средствам атрибута `for`.

3. `input` – позволяет создавать разные элементы интерфейса форм (см. таблицу 11)

4. `select` – создает список, элементы списка задаются с помощью тегов `option`.

Таблица 9 – Пример использования `select`

HTML-код	Результат

`<label for="topic">Тема</label>`
`<select name="topic" id="topic">`
`<option value="1">`
`Заказать услугу`
`</option>`
`<option value="2">`
`Оставить отзыв/замечание`
`</option>`
`<option value="3">`
`Предложить сотрудничество`
`</option>`
`<option value="4">`
`Другое`
`</option>`
`</select>`
	

Обратите внимание, что значения атрибутов `for` метки `label` и `id` у тега `select` совпадают, это необходимо для связи метки с элементом. При щелчке мыши на метке, активным становится связанный с ним элемент интерфейса формы.

5. `textarea` – создает многострочное поле ввода. Атрибут `rows` задает количество строк в поле ввода, а атрибут `cols` – количество символов в строке.

Таблица 10 – Пример использования `textarea`

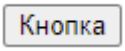
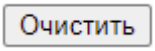
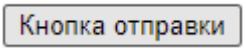
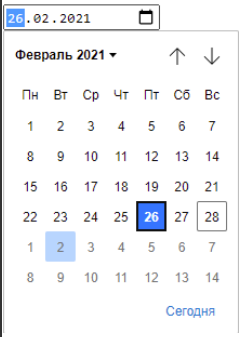
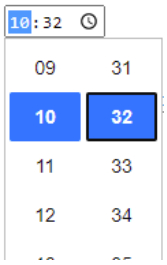
HTML-код	Результат

<label for="message">Текст сообщения</label> <textarea name="message" id="message" cols="30" rows="10"></textarea>	Текст сообщения

Большинство элементов интерфейса формы задаются с помощью незакрывающегося тега `input`. Тип элемента определяется атрибутом `type`. Существует более 20 различных типов `input`, в таблице представлены некоторые наиболее часто используемые.

Таблица 11 – Типы тега input

Значение type	**Описание**	**Пример**
text	Текстовое поле ввода	<input name="name" id="name" type="text"> Имя
checkbox	Флажки позволяют выбрать более одного варианта из предложенных	<label > <input id="check1" type="checkbox" checked> Чекбокс 1 </label> <label> <input id="check2" type="checkbox"> Чекбокс 2 </label> Чекбокс 1 Чекбокс 2
radio	Переключатели используются, когда следует выбрать один вариант из нескольких предложенных	<label> <input type="radio" name="option" id="opt1" value="option1" chacked>Радиокнопка 1 </label> <label> <input type="radio" name="option" id="opt2" value="option2" disabled>Радиокнопка 2 </label> Радиокнопка 1 Радиокнопка 2
file	Поле для ввода имени файла, который пересылается на сервер	<label for="loadfile"> Загрузить файл </label> <input type="file" id="loadfile">

		Загрузить файл Выберите файл Файл не выбран
button	Кнопка	`<button type="submit">Кнопка</button>` 
reset	Кнопка для возвращения данных формы в первоначальное значение	`<input type="reset" value="Очистить">` 
submit	Кнопка для отправки данных формы на сервер	`<input type="submit" value="Кнопка отправки">` 
hidden	Скрытое поле не отображается на веб-странице. Используется для отправки скрытых от пользователя данных.	`<input name="id_user" id="id_user" type="hidden" value="user1">`
password	Поле для ввода пароля	`<input name="password" id="password" type="password">` 
date	Поле для выбора календарной даты	`<input name="date" id="date" type="date">` 
time	Поле для выбора времени	`<input name="time" id="time" type="time">` 
email	Поле для ввода адреса электронной почты	`<input name="name" id="name" type="email">` 

number	Поле для ввода только чисел	`<input name="age" id="age" type="number">` Возраст 7
range	Ползунок для выбора чисел в указанном диапазоне	`<input name="age" id="age" type="range" min="0" max="100" step="1" value="18">` Возраст
tel	Поле для ввода телефонных номеров	`<input name="password" id="password" type="email">` Телефон

Валидация формы – это проверка корректности заполнения данных в форме. Для валидации форм часто используется язык программирования JavaScript, но базовые возможности проверки корректности заполнения формы заложены в HTML5 и CSS3.

У большинства элементов форм есть атрибут `required` без параметра. Добавление этого атрибута делает заполнение элемента обязательным. Например, зададим простую форму:

```
<form id="contact-form" action="">
      <label for="name">Имя</label><br>
      <input name="name" id="name" type="text" required >
      <input type="submit">
</form>
```

При попытке нажать на кнопку, браузер будет выдавать сообщения, а отправка данных прерываться.

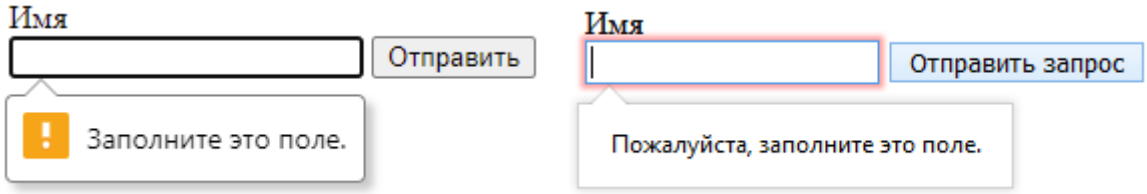


Рисунок 44 – Валидация формы в браузере Google Chrome (слева) и Firefox Mozilla (справа)

Также можно использовать паттерны заполнения. Паттерн – это шаблон формата ввода текстовых полей. Паттерны основаны на регулярных выражениях.

Паттерны задаются в атрибуте `pattern` у тестовых полей. Пример паттерна для телефона в формате +7-xxx-xxx-xx-xx:

```
pattern="\+7\-[0-9]{3}\-[0-9]{3}\-[0-9]{2}\-[0-9]{2}"
```

Пример паттерн для даты в формате дд.мм.гггг:

```
pattern="[0-9]{2}\.[0-9]{2}\.[0-9]{4}"
```

Такие типы `input`-ов как `date`, `time`, `email`, `number`, `tel` и подобные содержат встроенные паттерны проверки ввода данных.

Помимо стандартного браузерного оформления полей, прошедших и непрошедших проверку, можно использовать собственные стили для полей формы. В CSS существует четыре специальных псевдо класса, применимых к полям формы:

- `:valid` (валидное поле),
- `:invalid` (невалидное),
- `:required` (обязательное),
- `:optional` (необязательное).

Их можно использовать, чтобы добавлять некоторые подсказки пользователям, заполняющим форму.

Пример:

```
input:invalid {
    background: rgba(128,0,0,0.2);
   }
input:valid {
    background: rgba(0,128,0,0.2);
}
```

Для форм помимо псевдо классов очень важны селекторы атрибутов. Большинство элементов формы задаются с помощью тега `input`, при этом вид разных типов `input` очень отличается и для каждого типа есть необходимость задавать собственные стили. Это легко сделать через селекторы атрибутов. Пример:

```
input[type="text"]{
…
}
input[type="submit"]{
…
}
input[type="submit"]:hover{
…
}
```

1.10. Практическая работа 4. Таблицы и формы

Цель работы: добавить на страницы сайта таблицы и формы.

Задачи работы:

1. Изучить теги и CSS-правила для оформления таблиц.
2. Создать таблицу простой структуры.
3. Изучить теги для создания форм.
4. Создать простую форму.
5. Осуществить проверку формы на клиентской стороне – проверить заполнены ли обязательные поля, соответствуют ли введеные данные заданному формату.

Инструменты: Visual Studio Code, браузер Google Chrome.

Задание 1. Создание таблицы на странице Услуги

На странице услуги создадим таблицу прайс-листа как на рисунке 45.

Прайс-лист

#	Услуга	Описание	Цена
1	Услуга 1	Следует отметить, что рамки и место обучения кадров позволяет оценить значение представляет собой интересный эксперимент новых принципов формирования материально-технической и кадровой базы.	9 500 руб.
2	Услуга 2	Следует отметить, что рамки и место обучения кадров позволяет оценить значение представляет собой интересный эксперимент новых принципов формирования материально-технической и кадровой базы.	9 500 руб.
3	Услуга 3	Следует отметить, что рамки и место обучения кадров позволяет оценить значение представляет собой интересный эксперимент новых принципов формирования материально-технической и кадровой базы.	9 500 руб.
4	Услуга 4	Следует отметить, что рамки и место обучения кадров позволяет оценить значение представляет собой интересный эксперимент новых принципов формирования материально-технической и кадровой базы.	9 500 руб.

Рисунок 45 – Таблица на странице услуги

1. Создайте новую секцию на странице услуг для прайс-листа.
2. В этой секции создайте таблицу с заголовком и телом. В таблице будут 4 столбца:

```
<table>
    <thead>
        <tr>
            <th>#</th>
            <th>Услуга</th>
            <th>Описание</th>
            <th>Цена</th>
        </tr>
    </thead>
    <tbody>
        ...
    </tbody>
</table>
```

3. Заполните таблицу 4 – 6 строками тестовых данных.

4. У тега `table` задайте класс `price`.

5. Задайте стили для оформления рамок таблицы. Сначала задайте рамку для тега `table`:

```
table {
    border:5px solid #f60ad5;
}
```

#	Услуга	Описание	Цена
1	Услуга 1	Следует отметить, что рамки и место обучения кадров позволяет оценить значение представляет собой интересный эксперимент новых принципов формирования материально-технической и кадровой базы.	9 500 руб.
2	Услуга 2	Следует отметить, что рамки и место обучения кадров позволяет оценить значение представляет собой интересный эксперимент новых принципов формирования материально-технической и кадровой базы.	9 500 руб.
3	Услуга 3	Следует отметить, что рамки и место обучения кадров позволяет оценить значение представляет собой интересный эксперимент новых принципов формирования материально-технической и кадровой базы.	9 500 руб.
4	Услуга 4	Следует отметить, что рамки и место обучения кадров позволяет оценить значение представляет собой интересный эксперимент новых принципов формирования материально-технической и кадровой базы.	9 500 руб.

Рисунок 46 – Рамка у тега `table`

6. Задайте рамку у ячеек таблицы:

```
td, th {
    border: 2px solid white;
}
```

#	Услуга	Описание	Цена
1	Услуга 1	Следует отметить, что рамки и место обучения кадров позволяет оценить значение представляет собой интересный эксперимент новых принципов формирования материально-технической и кадровой базы.	9 500 руб.
2	Услуга 2	Следует отметить, что рамки и место обучения кадров позволяет оценить значение представляет собой интересный эксперимент новых принципов формирования материально-технической и кадровой базы.	9 500 руб.
3	Услуга 3	Следует отметить, что рамки и место обучения кадров позволяет оценить значение представляет собой интересный эксперимент новых принципов формирования материально-технической и кадровой базы.	9 500 руб.
4	Услуга 4	Следует отметить, что рамки и место обучения кадров позволяет оценить значение представляет собой интересный эксперимент новых принципов формирования материально-технической и кадровой базы.	9 500 руб.

Рисунок 47 – Рамки у ячеек таблицы

7. Чтобы схлопнуть рамки, добавьте к стилям таблицы правило `border-collapse: collapse;`

#	Услуга	Описание	Цена
1	Услуга 1	Следует отметить, что рамки и место обучения кадров позволяет оценить значение представляет собой интересный эксперимент новых принципов формирования материально-технической и кадровой базы.	9 500 руб.
2	Услуга 2	Следует отметить, что рамки и место обучения кадров позволяет оценить значение представляет собой интересный эксперимент новых принципов формирования материально-технической и кадровой базы.	9 500 руб.
3	Услуга 3	Следует отметить, что рамки и место обучения кадров позволяет оценить значение представляет собой интересный эксперимент новых принципов формирования материально-технической и кадровой базы.	9 500 руб.
4	Услуга 4	Следует отметить, что рамки и место обучения кадров позволяет оценить значение представляет собой интересный эксперимент новых принципов формирования материально-технической и кадровой базы.	9 500 руб.

Рисунок 48 – Схлопывание рамок

8. Определите относительную ширину колонок для ячеек заголовка таблицы с помощью псевдо классов:

```css
.price th:first-child{
    width: 10%;
}

.price th:nth-child(2), .price th:last-child{
    width: 25%;
}
```

#	Услуга	Описание	Цена
1	Услуга 1	Следует отметить, что рамки и место обучения кадров позволяет оценить значение представляет собой интересный эксперимент новых принципов формирования материально-технической и кадровой базы.	9 500 руб.
2	Услуга 2	Следует отметить, что рамки и место обучения кадров позволяет оценить значение представляет собой интересный эксперимент новых принципов формирования материально-технической и кадровой базы.	9 500 руб.
3	Услуга 3	Следует отметить, что рамки и место обучения кадров позволяет оценить значение представляет собой интересный эксперимент новых принципов формирования материально-технической и кадровой базы.	9 500 руб.
4	Услуга 4	Следует отметить, что рамки и место обучения кадров позволяет оценить значение представляет собой интересный эксперимент новых принципов формирования материально-технической и кадровой базы.	9 500 руб.

Рисунок 49 – Определение ширины столбцов

9. С помощью стилей задайте выравнивание текста по центру во всех ячейках (кроме ячеек с описанием услуг).

10. А также задайте во всех ячейках внутренние отступы по 10px.

#	Услуга	Описание	Цена
1	Услуга 1	Следует отметить, что рамки и место обучения кадров позволяет оценить значение представляет собой интересный эксперимент новых принципов формирования материально-технической и кадровой базы.	9 500 руб.
2	Услуга 2	Следует отметить, что рамки и место обучения кадров позволяет оценить значение представляет собой интересный эксперимент новых принципов формирования материально-технической и кадровой базы.	9 500 руб.
3	Услуга 3	Следует отметить, что рамки и место обучения кадров позволяет оценить значение представляет собой интересный эксперимент новых принципов формирования материально-технической и кадровой базы.	9 500 руб.
4	Услуга 4	Следует отметить, что рамки и место обучения кадров позволяет оценить значение представляет собой интересный эксперимент новых принципов формирования материально-технической и кадровой базы.	9 500 руб.

Рисунок 50 – Выравнивание и отступы

11. Оформите таблицу с помощью правил для рамок и заливки, а также используя псевдо классы (`nth-child()`) для чередующихся строк как на рисунке 51.

#	Услуга	Описание	Цена
1	Услуга 1	Следует отметить, что рамки и место обучения кадров позволяет оценить значение представляет собой интересный эксперимент новых принципов формирования материально-технической и кадровой базы.	9 500 руб.
2	Услуга 2	Следует отметить, что рамки и место обучения кадров позволяет оценить значение представляет собой интересный эксперимент новых принципов формирования материально-технической и кадровой базы.	9 500 руб.
3	Услуга 3	Следует отметить, что рамки и место обучения кадров позволяет оценить значение представляет собой интересный эксперимент новых принципов формирования материально-технической и кадровой базы.	9 500 руб.
4	Услуга 4	Следует отметить, что рамки и место обучения кадров позволяет оценить значение представляет собой интересный эксперимент новых принципов формирования материально-технической и кадровой базы.	9 500 руб.

Рисунок 51 – Оформление таблицы

12. Добавьте заголовок таблице – тег `caption`. И оформите его как на рисунке 45.

Задание 2. Создание страницы обратной связи

Примерный итоговый вид страницы обратной связи представлен на рисунке 52.

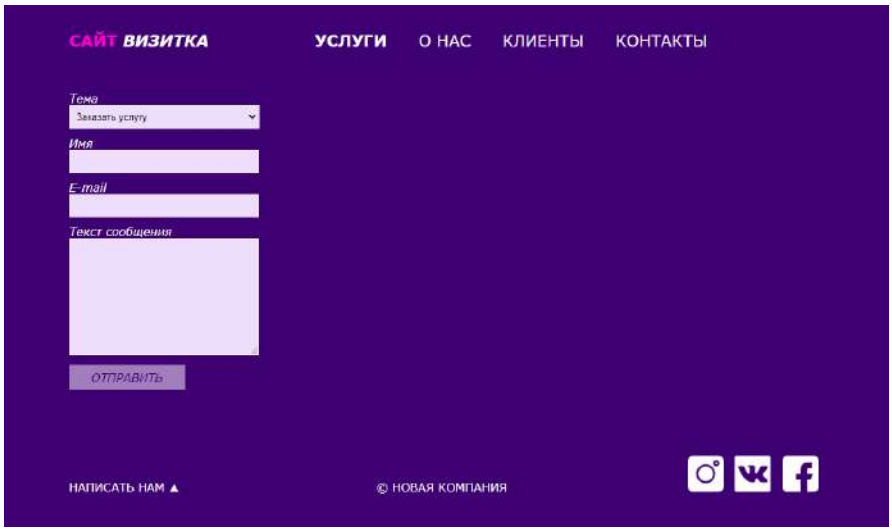


Рисунок 52 – Страница обратной связи

1. Создайте новую страницу – form.html. Оформите шапку и подвал сайта как на всех страницах (рис. 53).

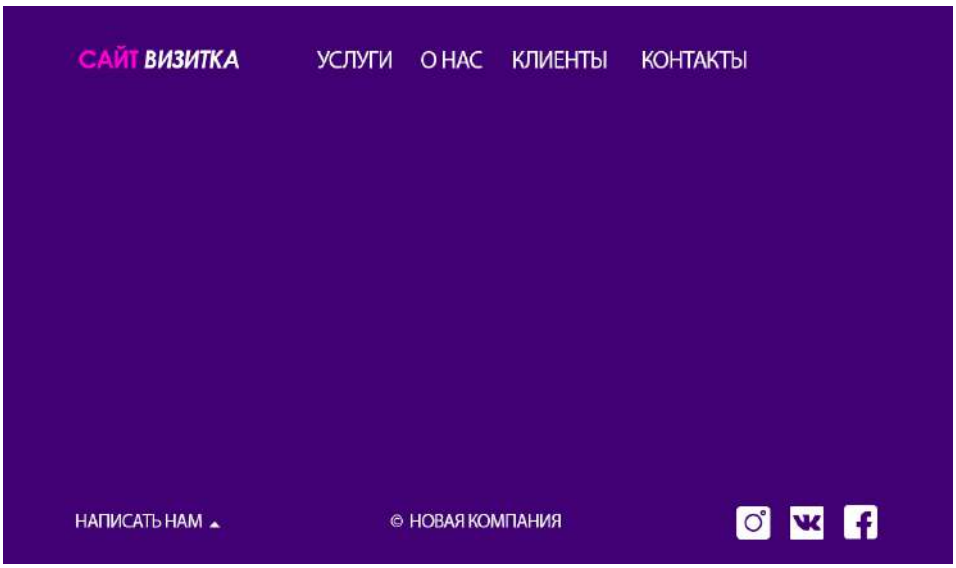


Рисунок 53 – Оформление шапки и подвала страницы

2. На всех страницах сайта текст *Написать нам* сделайте ссылками на страницу form.html.

3. В контейнере `main` создайте форму как на рисунке 54. Форма состоит из трех `input`-ов, два `input`-а с атрибутом `type="text"` (поле ввода), а последний с `type="submit"` (кнопка отправки данных на сервер).

Рисунок 54 – Форма из трех элементов

4. Каждый из `input`-ов оберните блоками `div` с классом `form__item`. Блоки формы теперь располагаются друг за другом по вертикали.

```
<form action="">
    <div class="form_item">
        <input type="text">
    </div>
    <div class="form_item">
        <input type="text">
    </div>
    <div class="form_item">
        <input type="submit">
    </div>
</form>
```

5. Для первых двух `input`-ов добавьте подписи с помощью тега `label`. Обратите внимание значение атрибута `for` у `label` и значение атрибута `id` у

input должны совпадать. Это осуществляет связь между подписью и элементом формы. Пример оформления подписи для первого поля ввода:

```
<div class="form__item">
    <label for="name">Имя</label><br>
    <input name="name" id="name" type="text">
</div>
```

6. Добавьте в начало формы новый элемент – поле выбора select:

```
<div class="form__item">
    <label for="topic">Тема</label>
    <select name="topic" id="topic">
        <option value="1">Заказать услугу</option>
        <option value="2">Оставить отзыв/замечание</option>
        <option value="3">Предложить сотрудничество</option>
        <option value="4">Другое</option>
    </select>
</div>
```

7. Самостоятельно добавьте многострочное поле ввода до кнопки отправки формы.

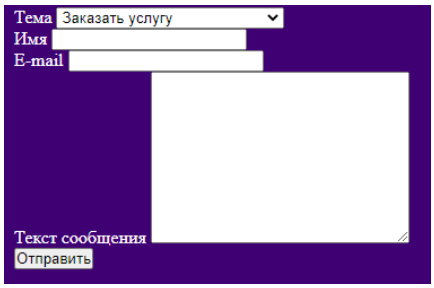


Рисунок 55 – Форма обратной связи до оформления

8. С помощью тега br разнесите подписи и элементы формы на разные строки (рис. 56).

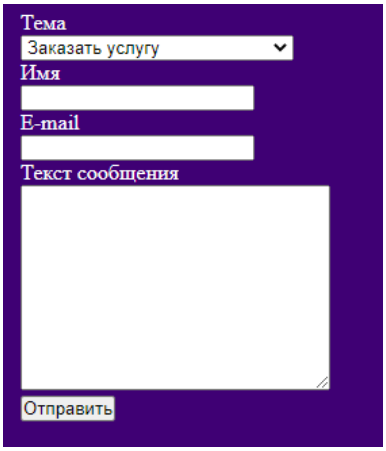


Рисунок 56 – Изменение расположения подписей

9. Задайте для формы (для тега `form`) идентификатор `contact-form`.

10. Задайте стилевые правила для input-ов ввода текста:

```
#contact-form input[type="text"]{
    height: 20px;
    width: 250px;
    padding: 5px;
    margin-bottom: 10px;
    border: white solid 1px;
    background-color: #eddfff;
}
```

Обратите внимание как задан селектор по атрибутам в правиле.

11. Оформите кнопку с помощью селектора `#contact-form input[type="submit"]` так, чтобы она выглядела как на рисунке 57.

Рисунок 57 – Вид кнопки

12. Добавьте стилевое правило для вида кнопки при наведении мыши – `#contact-form input[type="submit"]:hover`

Рисунок 58 – Вид кнопки при наведении

13. Оформите текстовое поле, поле выбора и подписи так, чтобы форма выглядела как на рисунке 59.

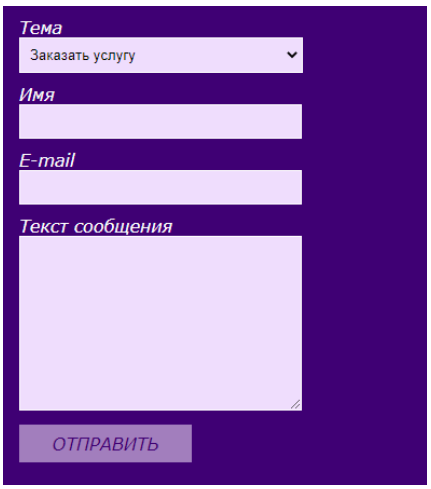


Рисунок 59 – Оформленная форма

Задание 3. Валидация формы

1. Для полей *Имя, E-mail* и *Текст сообщения* с помощью атрибута `placeholder` задайте подсказывающий текст. Подсказывающий текст – это текст, который отображается внутри поля формы, этот текст исчезает при получении элементом фокуса.
2. Для всех элементов формы, кроме кнопки задайте атрибут без значения `required`. Этот атрибут указывает, что поле обязательно для заполнения.
3. Попробуйте нажать кнопку *Отправить* не заполняя поля, частично заполняя поля и полностью заполнив все поля.
4. Попробуйте заполнить поле *E-mail* некорректным адресом. Пройдет ли проверку текущее поле?
5. Измените у поля *E-mail* тип на *email*. Не забудьте задать стили для этого типа `input`.
6. Попробуйте заполнить поле *E-mail* некорректным адресом. Пройдет ли проверку текущее поле?
7. Рассмотрите другие типы `input`-ов.
8. Добавьте в форму несколько полей, например, `date, tel` и др.

Глава 2 Верстка сайтов

2.1 Каркас сайта. Модульная сетка

Любая веб-страница состоит из укрупненных структурных блоков, например, как на рисунке 60:

– Шапка (header);
– Основная часть (main);
– Боковые панели (sidebar);
– Подвал (footer).

Рисунок 60 – Примерная структура страницы сайта

В свою очередь эти блоки делятся на более мелкие блоки. Внутри блоков располагаются различные элементы (текст, изображения, ссылки, видео и т.д.).

Модульная сетка (в веб-дизайне) представляет собой единую схему расположения всех блоков и элементов сайта. Модульная сетка представляет собой некоторый каркас, который проходит через всю страницу и определяет расположение всех блоков и элементов на ней.

В веб-дизайне чаще всего используется колоночная модульная сетка (рис. 61). Обычно сетка состоит из 12-ти колонок.

Рисунок 61 – Двенадцати-колоночная модульная сетка

При этом вся рабочая область страницы разбивается на 12 колонок и структурные блоки страницы имеют ширину, кратную числу колонок. Например, ширина в 1 колонку, в 2 колонки и т.д. до 12 колонок.

Колонки в веб-дизайне могут иметь конкретную абсолютную ширину, заданную в пикселях, а могут иметь ширину относительную, при этом ширина колонок меняется с изменением ширины окна браузера.

Существует три основных способа построения сеток:

1. Тег `div` с CSS-свойствами `display - block, inline, inline-block`.
2. Одномерные гибкие блоки `flexbox`.
3. Система двумерных сеток `grid layout`.

Первый способ был рассмотрен в практических работах первой главы пособия. Подробно рассмотрим второй способ построения сеток.

2.2 Flex-контейнеры

Одним из современных способов позиционирования блоков на странице является использование гибких блоков (flexbox). Для того чтобы простые блоки стали гибкими, достаточно у их **родителя** задать свойство `display:flex`.

Рассмотрим основные свойства, которые можно задавать у flex-контейнера (родительского элемента гибких блоков).

У flex-контейнера есть две оси – главная ось (по умолчанию направлена слева направо и поперечная ось – по умолчанию направлена сверху вниз). Блоки внутри flex-контейнера располагаются вдоль главной оси (рис. 62).

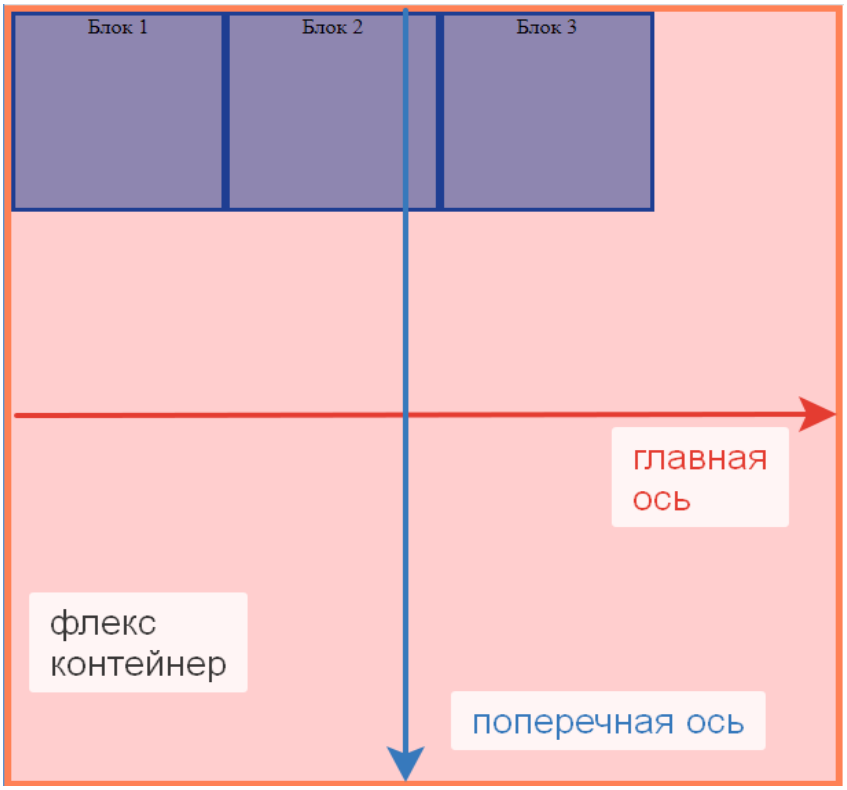


Рисунок 62 – Flex-контейнер (направление осей по умолчанию)

Изменить направление главной оси, а, следовательно, и изменить порядок вывода блоков внутри flex-контейнера, можно с помощью CSS-свойства **flex-direction.** Возможные значения:

- `flex-direction:row;` (по умолчанию слева направо);
- `flex-direction:column;` (сверху вниз, рис. 63);
- `flex-direction:row-reverse;` (справа налево);
- `flex-direction:column-reverse;` (снизу вверх).

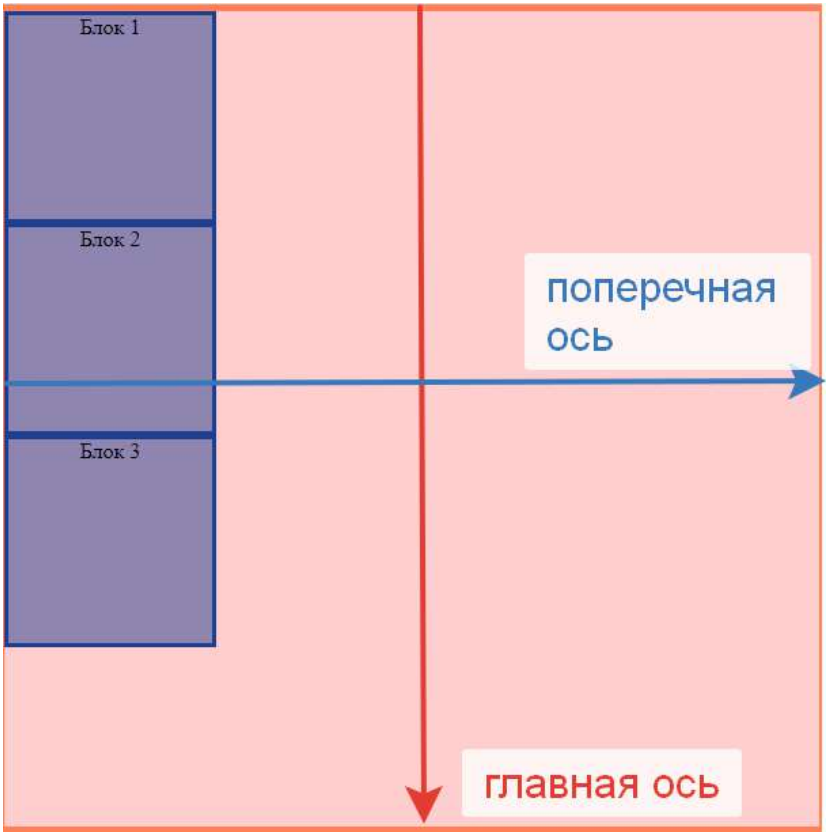


Рисунок 63 – Изменение направления главной оси

Поперечная ось используется при выравнивании элементов. Если главная ось направлена горизонтально, то поперечная ось смотрит вниз. Если главная ось направлена вертикально, то поперечная ось смотрит направо. Направление поперечной оси изменяется вместе с изменением главной оси. Поперечная ось не может быть направлена вверх или влево.

Выравнивание по главной оси задается с помощью CSS-свойства **justify-content.** Возможные значения (рис. 64):

- `flex-start`,
- `flex-end`,
- `center`,
- `space-between`,
- `space-around`.

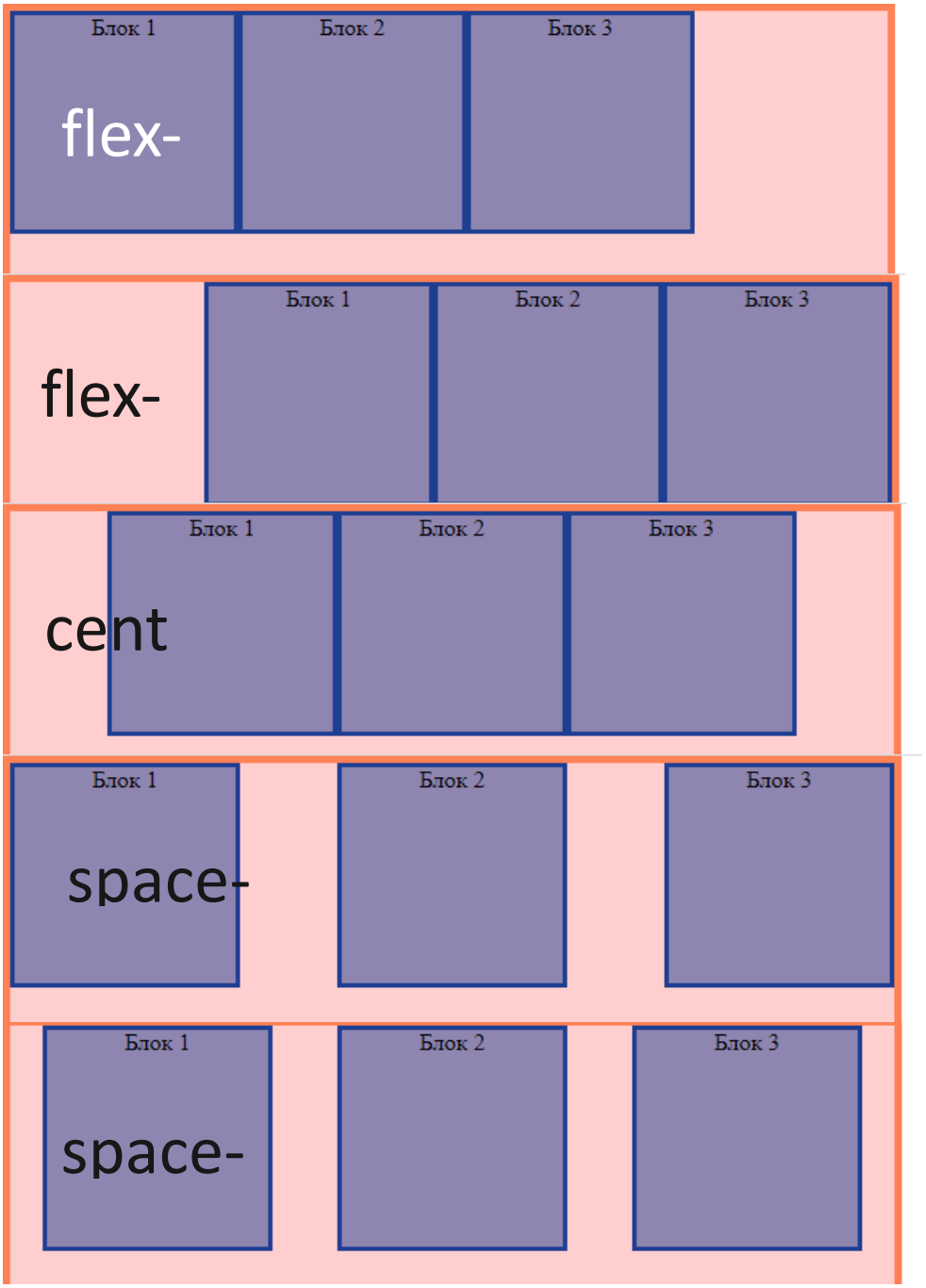


Рисунок 64 – Выравнивание по главной оси

Выравнивание вдоль поперечной оси осуществляется с помощью CSS-свойство **align-items**. Возможные значения:

– `stretch` (значение по умолчанию, если у flex-элементов не задана высота, то при этом свойстве элементы приобретают высоту родителя – растягиваются (рис. 65), если высота задана, то элементы выравнивается в начале поперечной оси);

– `flex-start` (выравнивание элементов в начале поперечной оси);

– `flex-end` (выравнивание элементов в конце поперечной оси);

– `center` (выравнивание элементов по центру поперечной оси);

– `baseline` (выравнивание элементов по базовой линии flex-элементов – рисунок 66).

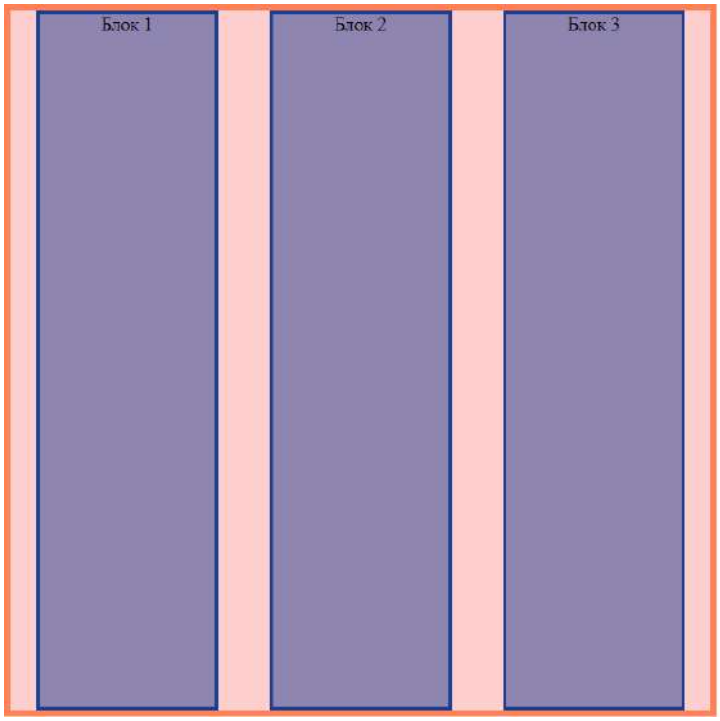


Рисунок 65 – Значение выравнивания `stretch` (у блоков не задана высота)

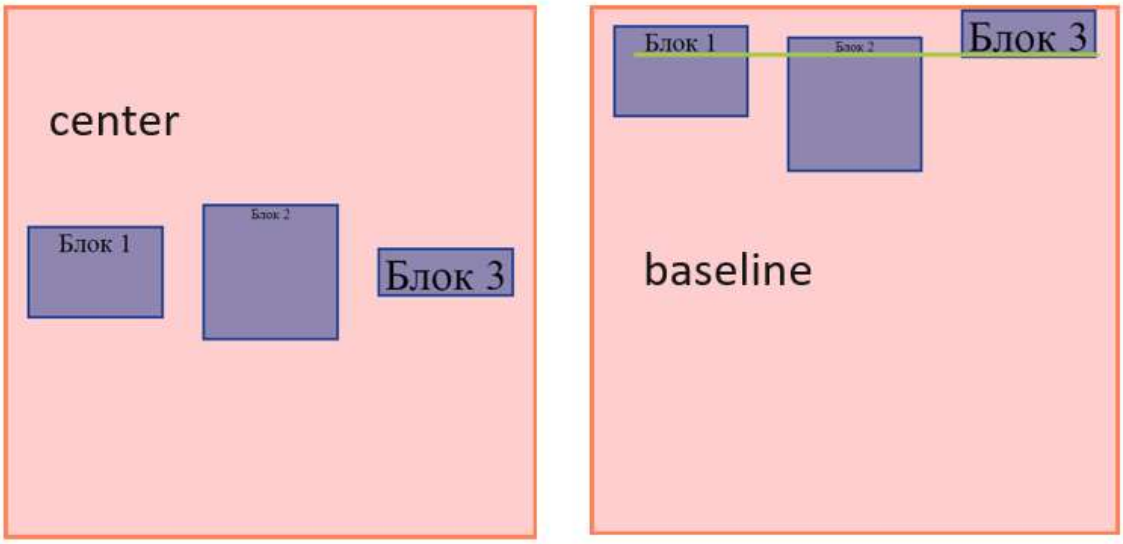


Рисунок 66 – Выравнивание по центру (слева) и выравнивание по базовой линии (справа)

Следующее свойство связано с переносом элементов. Сначала определим поведение элементов внутри flex-контейнера. Если суммарная ширина элементов меньше ширины контейнера, то они располагаются внутри контейнера, не меняя своей ширины (рис. 67).

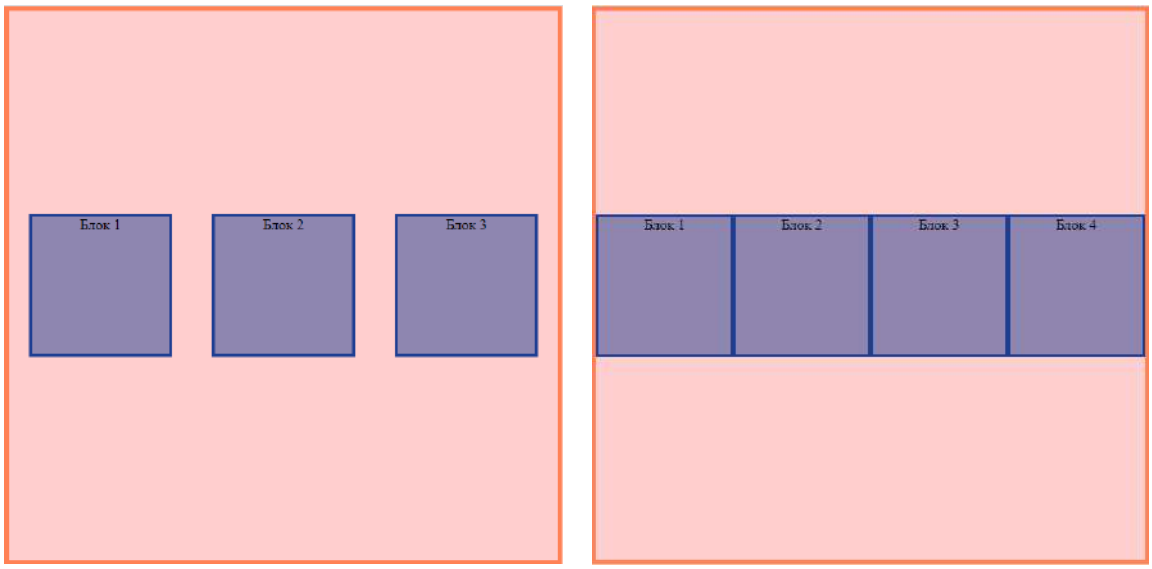


Рисунок 67 – Расположение элементов внутри контейнера

Если суммарная ширина элементов больше ширины родителя, то элементы сужаются (рис.68).

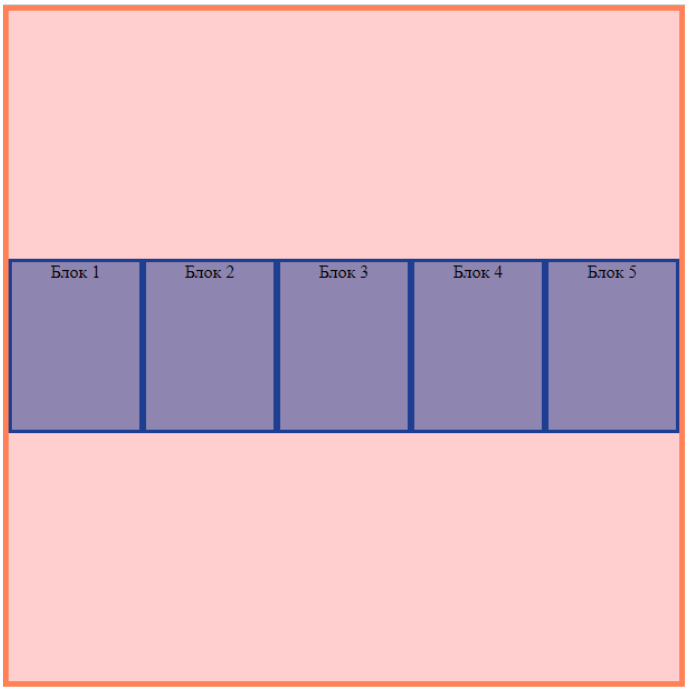


Рисунок 68 – Расположение элементов внутри контейнера (сужение ширины элементов)

Если содержимое элементов не позволяет им сузиться под ширину родителя, то происходит переполнение контейнера и элементы выходят за его пределы (рис 69).

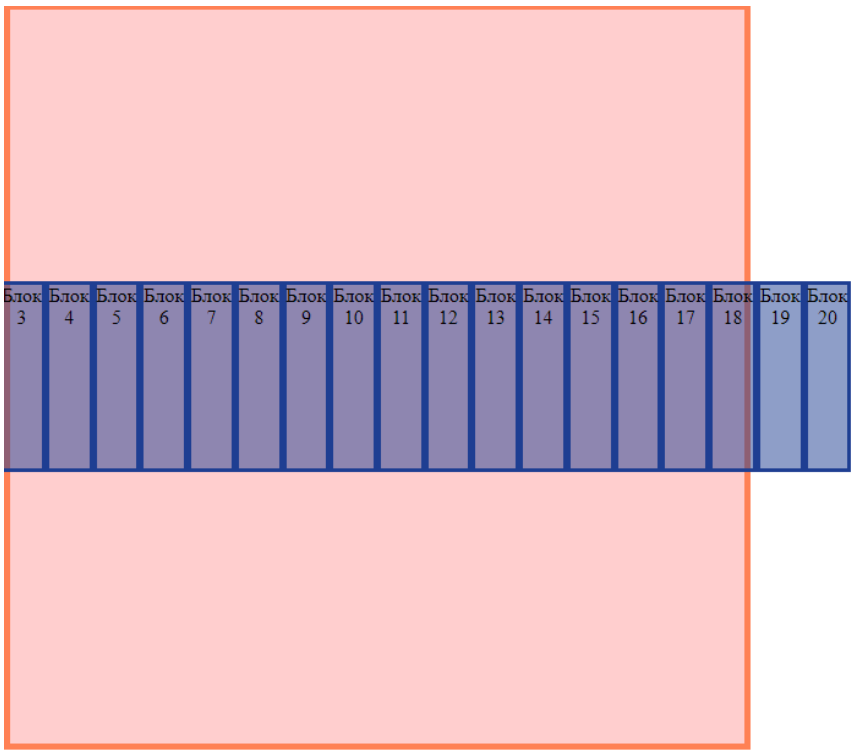


Рисунок 69 – Расположение элементов внутри контейнера (переполнение)

Поведение элементов при переполнении можно поменять с помощью CSS-свойства **flex-wrap.** Возможные значения

– `nowrap` – перенос flex-элементов на новую строку запрещён (значение по умолчанию).

– `wrap` – разрешает перенос flex-элементов на новую строку. Ряды элементов располагаются вдоль поперечной оси, первый ряд – в начале поперечной оси, последний – в конце (рис. 70).

– `wrap-reverse` – разрешает перенос flex-элементов на новую строку. Ряды элементов располагаются в обратном порядке: первый – в конце поперечной оси, последний – в начале (рис. 71).

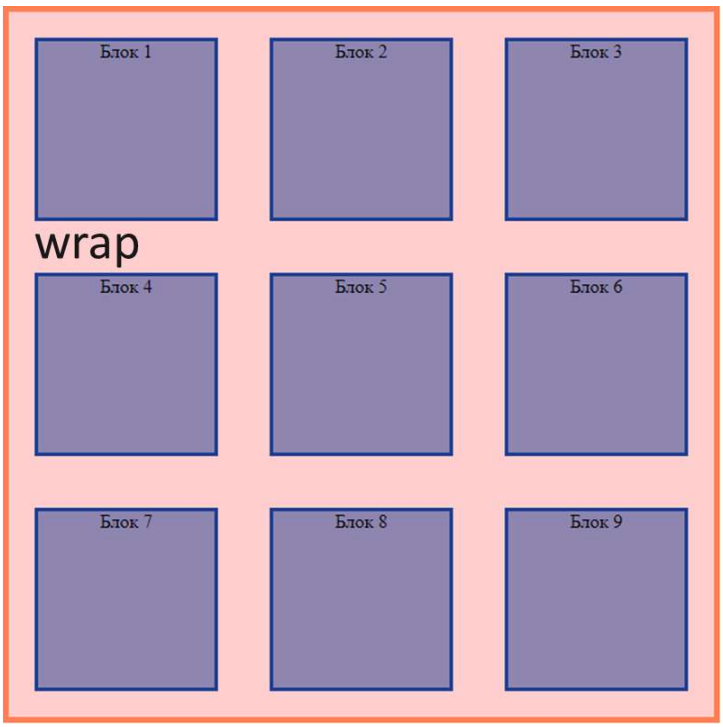


Рисунок 70 – Перенос элементов (`flex-wrap:wrap`)

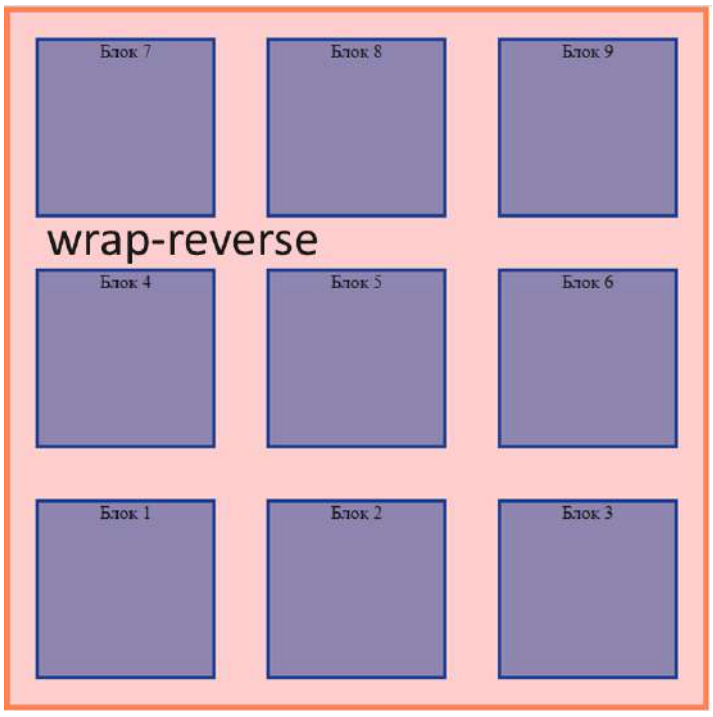


Рисунок 71 – Перенос элементов (`flex-wrap:wrap-reverse`)

Для выравнивания вдоль поперечной оси нескольких строк блоков используется свойство CSS-свойство **align-content**. Возможные значения свойства:

– `stretch` – значение по умолчанию, растягивает ряды flex-элементов, при этом оставшееся свободное место между ними делится поровну. Отображение строк при этом зависит от значения свойства `align-items`:
    - если у `align-items` задано значение `stretch`, то элементы в строках растягиваются на всю высоту своей строки.
    - если значение отлично от `stretch`, то элементы в строках ужимаются под своё содержимое и выравниваются в строках в зависимости от значения `align-items`.
– `flex-start`,
– `flex-end`,
– `center`,
– `space-between`,
– `space-around`.

Свойство `align-content` переопределяет заданное значение `align-items`.

Flexbox по своей сути является моделью одномерного макета. Flex-элементы внутри flex-контейнера можно размещать как горизонтально, так и вертикально, но не одновременно. Если вы хотите располагать элементы в двух направлениях, вам потребуется вставить один flex-контейнер в другой (рис. 72).

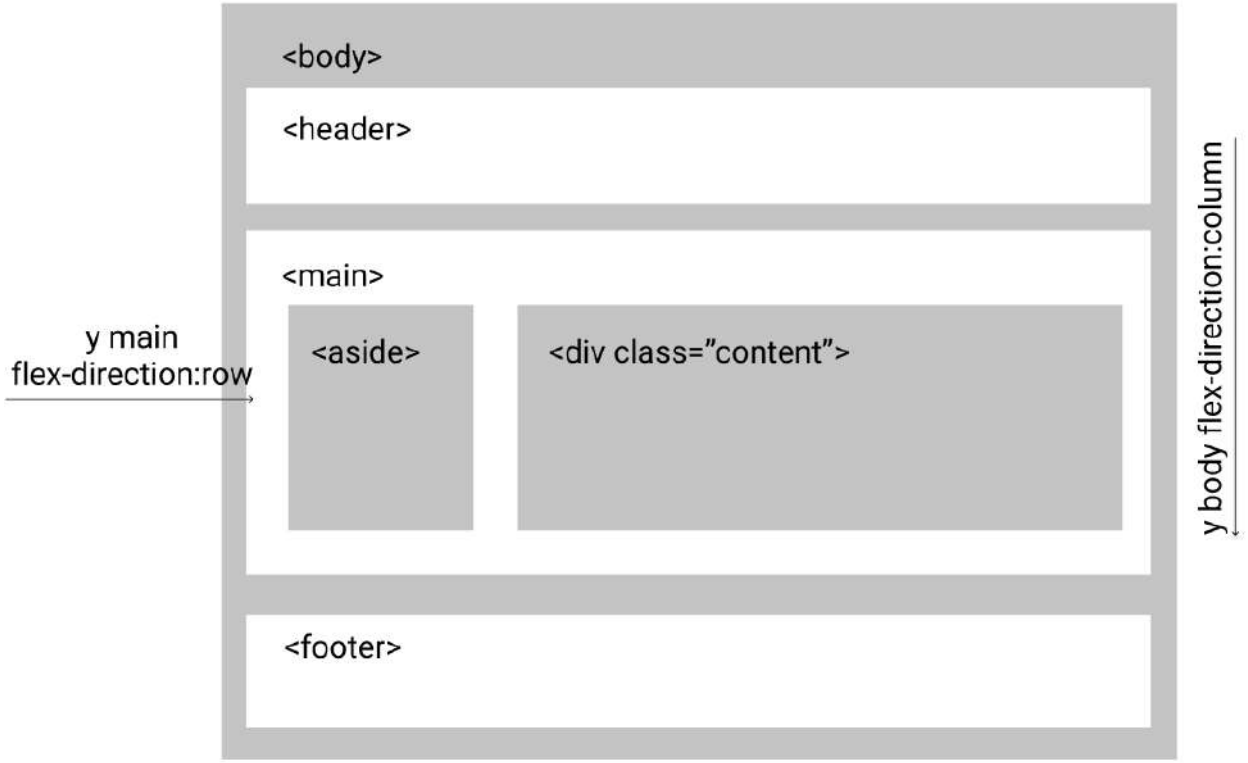


Рисунок 72 – Размещение flexbox

2.3 Flex-элементы

Flex-элементы имеют ряд отличий от обычных блочных элементов:

– flex-элементы не растягиваются на всю ширину контейнера по умолчанию;

– на flex-элементы не действует свойство float;

– внешние отступы не схлопываются ни по горизонтали, ни по вертикали;

– внешние отступы не выпадают ни из flex-контейнера, ни из flex-элементов.

При этом свойства `border` и `padding` у flex-элементов работают также, как и у блочных. Но `margin`, как и `padding` «не знают» понятия направление оси. Поэтому, когда главная ось направлена слева направо, горизонтальные отступы перемещают flex-элементы вдоль главной оси. Но если направить главную ось сверху вниз, то те же отступы начнут работать вдоль поперечной оси. Аналогично работают стили для высоты и ширины блоков:

– ширина width – размер блока по горизонтали всегда,

– высота height – размер блока по вертикали всегда.

У flex-элементов есть собственное свойство, которое задает размер элемента по главной оси – **flex-basis**. В зависимости от того, как направлена ось, это свойство может задавать ширину или высоту блока.

Следующее свойство отвечает за распределение свободного места между элементами внутри контейнера. Механизм распределения свободного места при использовании flex:

– находятся элементы, у которых есть внешние отступы со значением auto;

– всё свободное место в соответствующих направлениях отдаётся таким отступам (то есть задаётся определённый размер отступа в пикселях);

– если элементов с автоматическими отступами на одном направлении несколько, то место между ними перераспределяется поровну;

– только после этого запускаются механизмы выравнивания.

Рассмотрим важные свойства flex-элементов. Первое свойство – это порядок вывода элементов на страницу. По умолчанию у всех элементов на странице порядковый номер равен 0, и они выводятся в том же порядке, в каком расположены в коде.

CSS-свойство **order** меняет порядковый номер элемента. В качестве значения можно назначить положительное или отрицательное целое число.

Сортировка производится по возрастанию номера. Таким образом отрицательное число для свойства `order` смещает элемент вперед, а положительное – в конец.

CSS-свойство, отвечающее за растягивание блока – **flex-grow** принимает неотрицательные числовые значения, его значение по умолчанию – 0. Если значение `flex-grow` равно 0, то flex-элемент «не претендует» на оставшееся свободное место во flex-контейнере и не будет увеличиваться, чтобы занять это место. Если значение `flex-grow` больше нуля, то flex-элемент будет увеличиваться, «захватывая» оставшееся свободное место.

CSS-свойство **flex-shrink** принимает неотрицательные числовые значения, его значение по умолчанию – 1. Если значение `flex-shrink` больше нуля, то flex-элемент будет уменьшаться, «впитывая» часть отрицательного пространства, если оно существует. Если значение `flex-shrink` равно нулю, то flex-элемент уменьшаться не будет.

Подробный расчёт коэффициентов сжатия и растяжения можно посмотреть в источнике [6].

2.5 Практическая работа 5. 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. 73).

```
<> 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 73 – 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>
```

1. В папке css разместите файл нормализации стилей (normalize.css).
2. Подключите его к странице index.html.
3. В папке css создайте файл style.css.
4. Подключите этот файл стилей к странице index.html после подключения normalize.css.
5. 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;
}
```

6. Сохраните все файлы, просмотрите результат в браузере (рис. 74). 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.

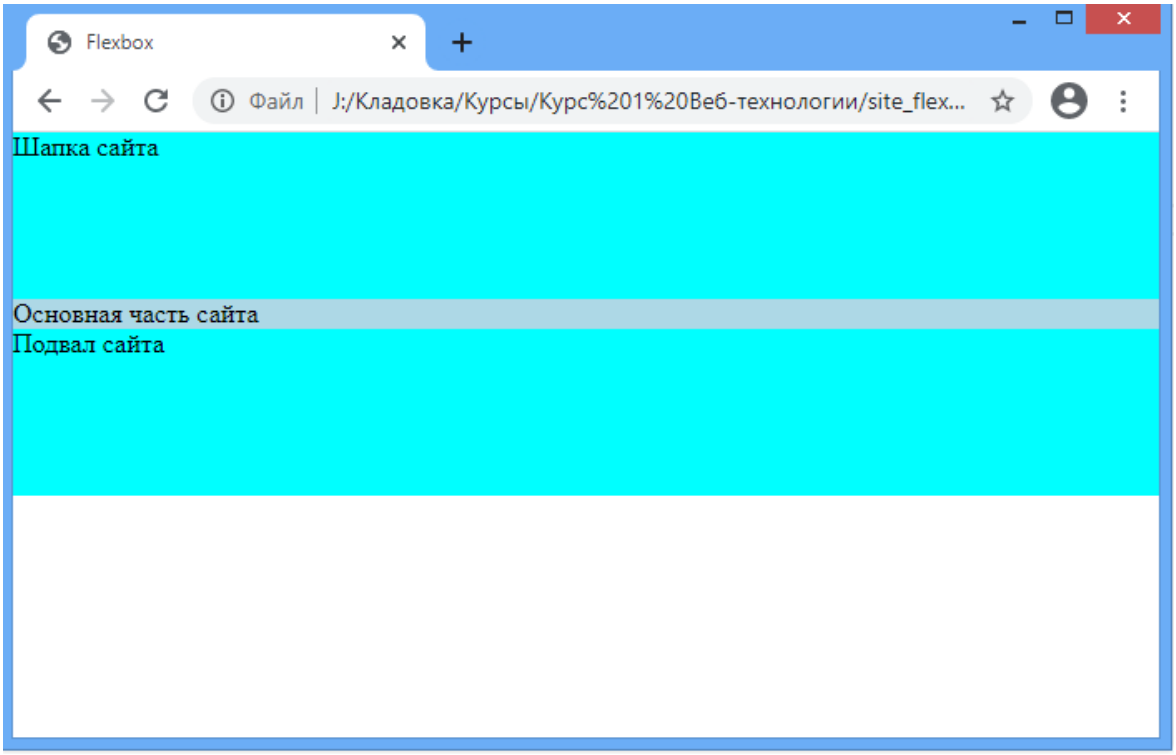


Рисунок 74 – Вид страницы в браузере

7. Сделайте контейнер с классом `container` flex-контейнером:

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

8. Просмотрите результат (рис. 75).

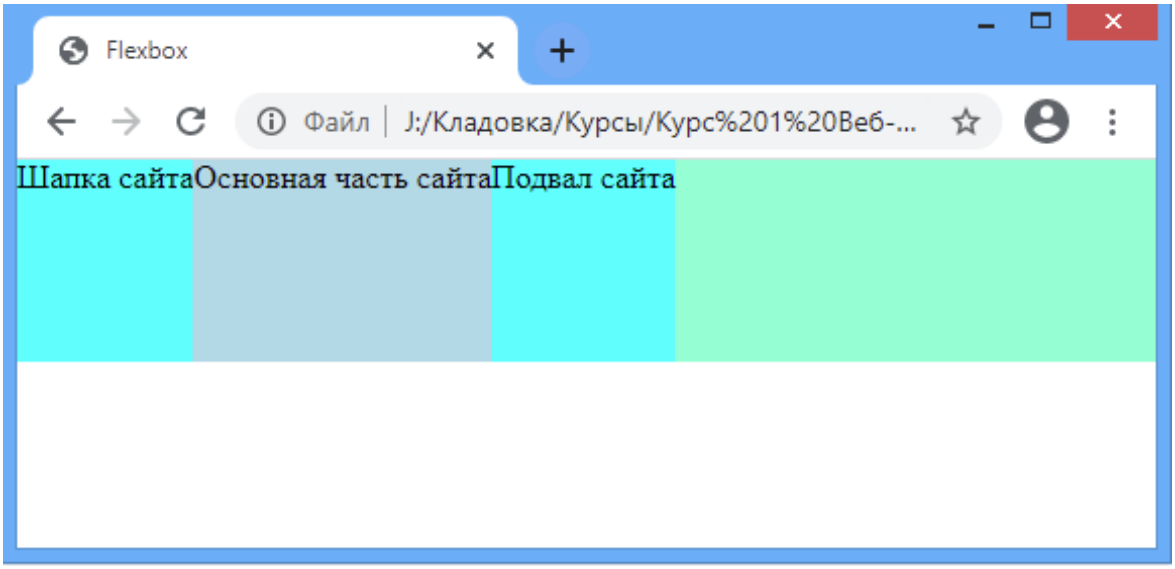


Рисунок 75 – Расположение flex-элементов по умолчанию

9. Измените направление вывода flex-элементов с помощью стилевого правила `flex-direction: column;` (сверху вниз). Просмотрите результат – сайт должен вернуться к исходному виду (рис. 74).

10. 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;
}
```

11. Просмотрите результат (рис. 76).

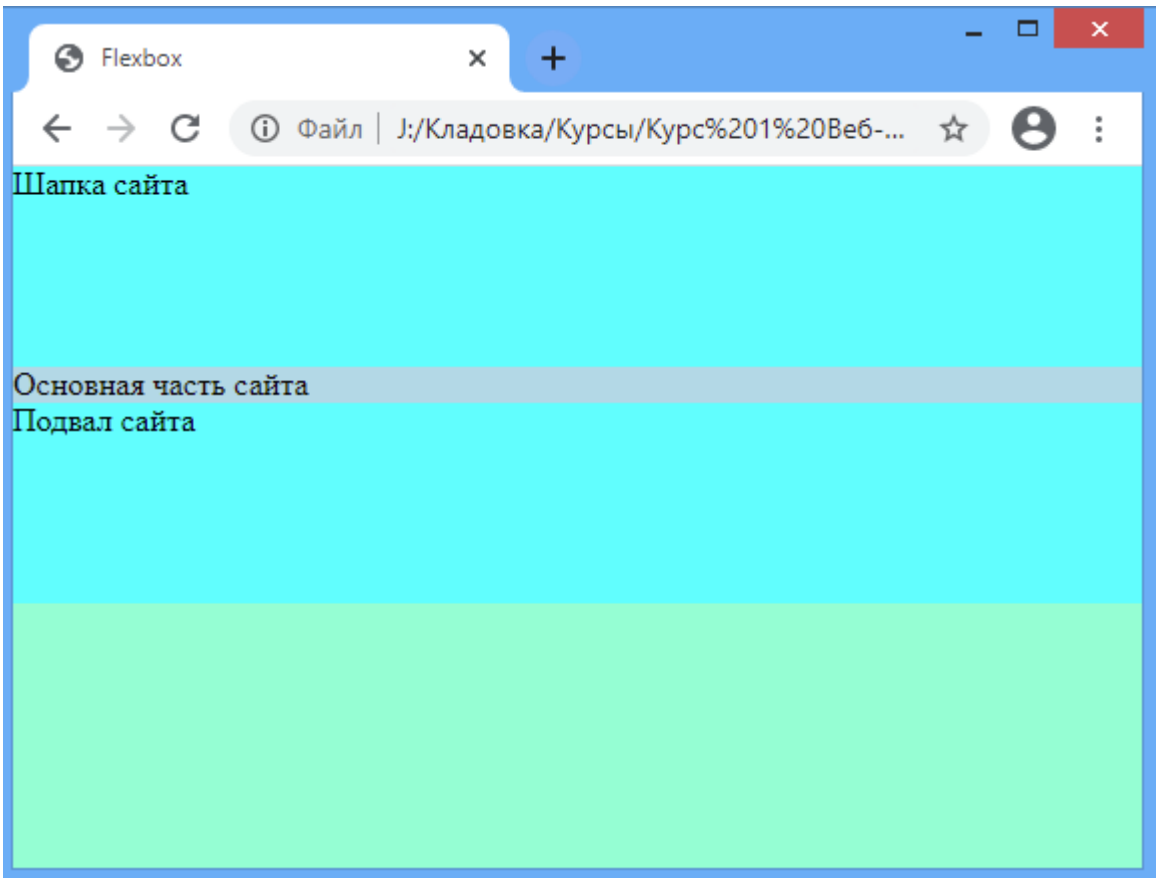


Рисунок 76 – Высота контейнера равна высоте окна браузера

12. 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;
}
```

13. Просмотрите результат работы (рис. 77)

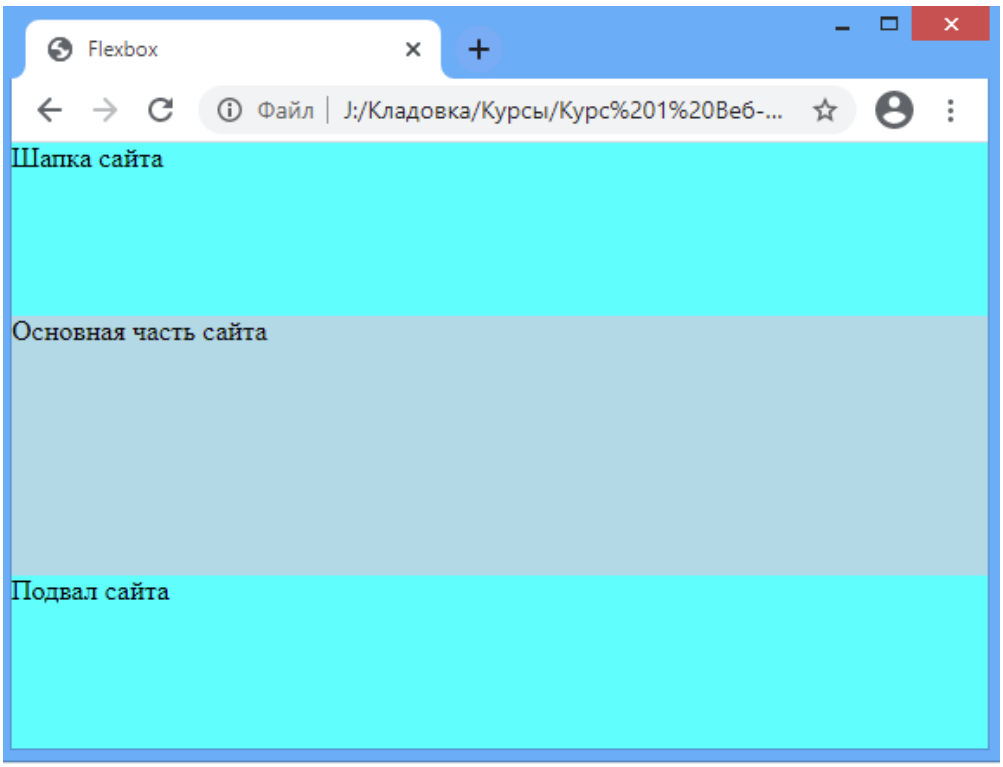


Рисунок 77 – Подвал прижат к низу окна

14. Сделайте контейнер `container` фиксированной ширины и центрированным по горизонтали (рис. 78). Задайте у него стилевые правила: ширина 1200px, внешние отступы сверху и снизу по 0, а справа и слева – `auto`.

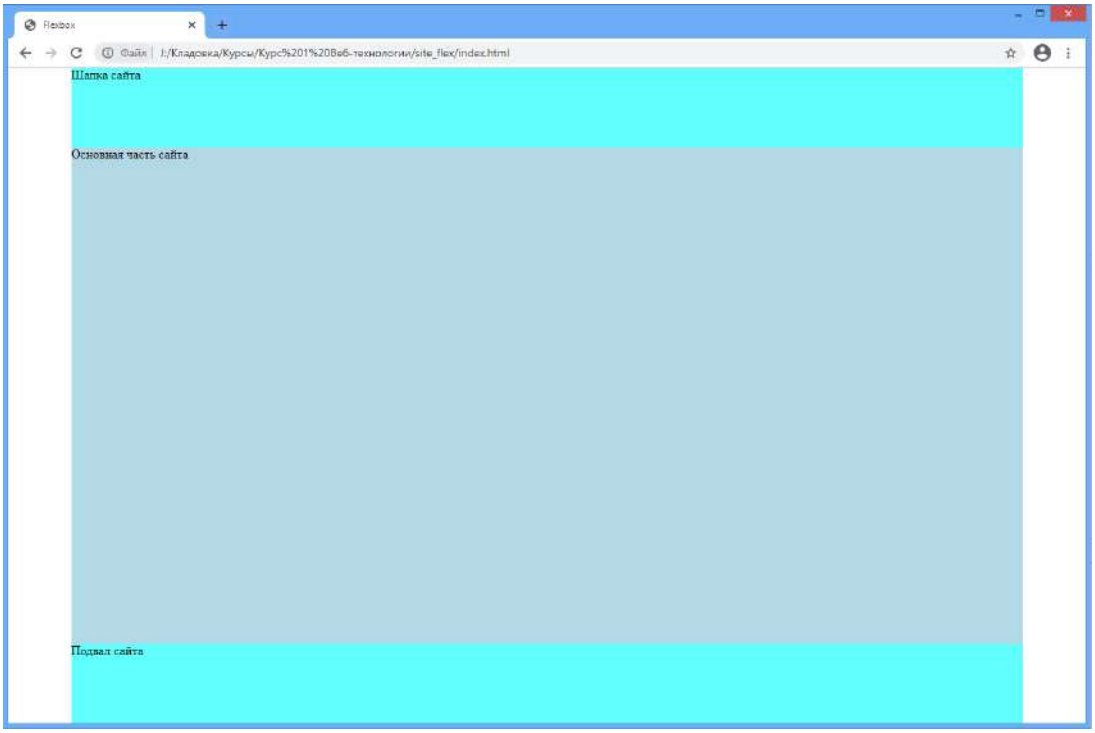


Рисунок 78 – Контейнер фиксированной ширины

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. 79).

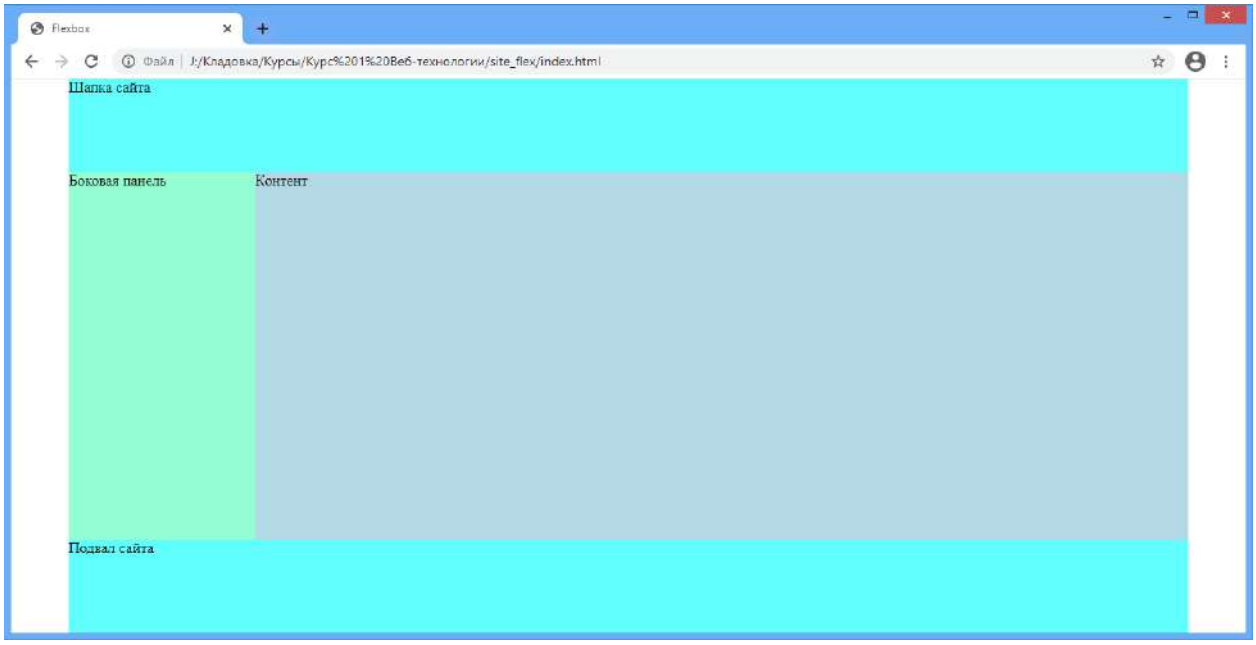


Рисунок 79 – Добавление боковой панели

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. 80). Оформите их с помощью стилей без использования flex-свойств.

Рисунок 80 – 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. 81).

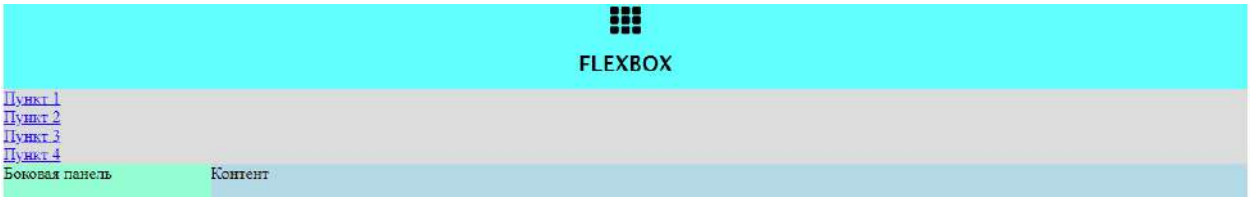


Рисунок 81 – Добавление меню

6. Измените с помощью стилей оформление пунктов меню (рис. 82).

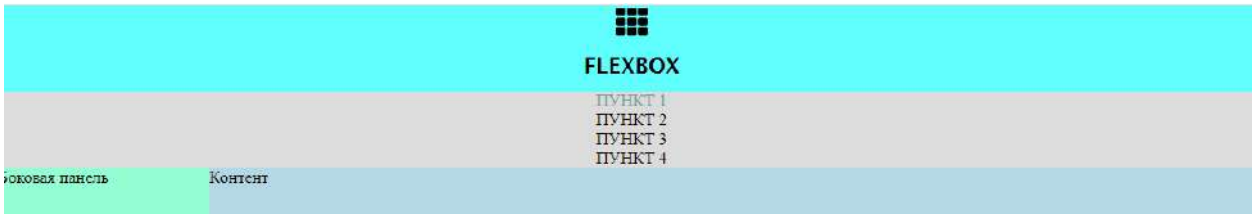


Рисунок 82 – Изменение внешнего вида меню

7. Сделайте блок с классом `menu` flex-контейнером и направлением основной оси – слева направо (рис. 83).

Рисунок 83 – Создание flex-меню

8. Задайте выравнивание блоков по горизонтали (рис. 84):

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

Рисунок 84 – Выравнивание пунктов по горизонтали

9. У блоков меню добавьте внутренние отступы, а у всего меню – внутренние отступы сверху и снизу (рис. 85).

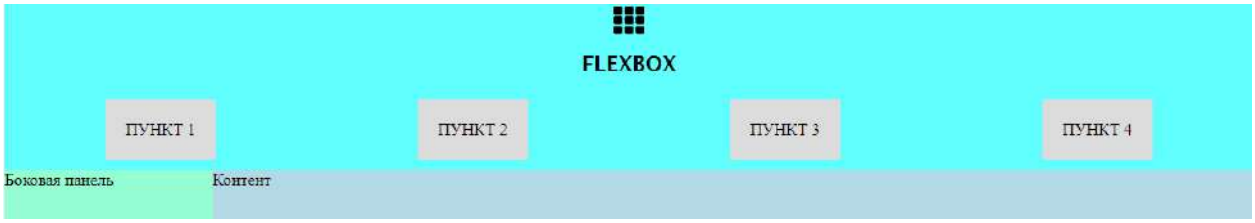


Рисунок 85 – Задание отступов

10. Добавьте еще пункты меню, чтобы они перестали вмещаться на одну строку (рис. 86).

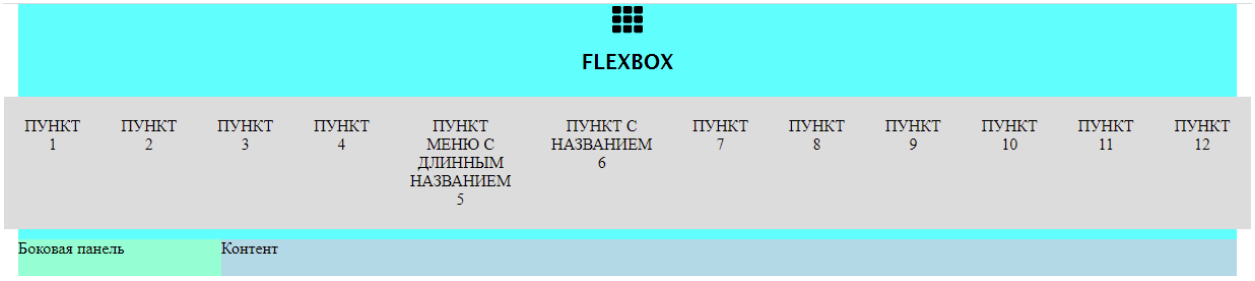


Рисунок 86 – Переполнение контейнера

11. Разрешите flex-контейнеру переносить блоки – стилевое правило `flex-wrap:wrap` (рис. 87).

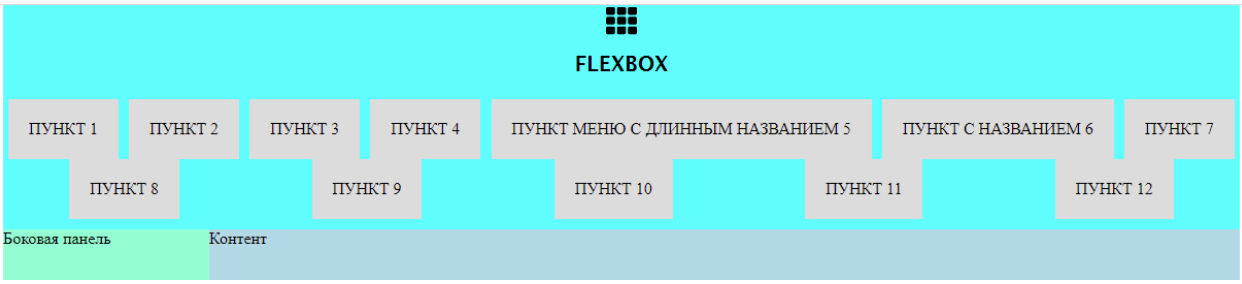


Рисунок 87 – Перенос пунктов меню

12. Разрешите flex-элементам (пунктам меню) расти в ширину (рис. 88).

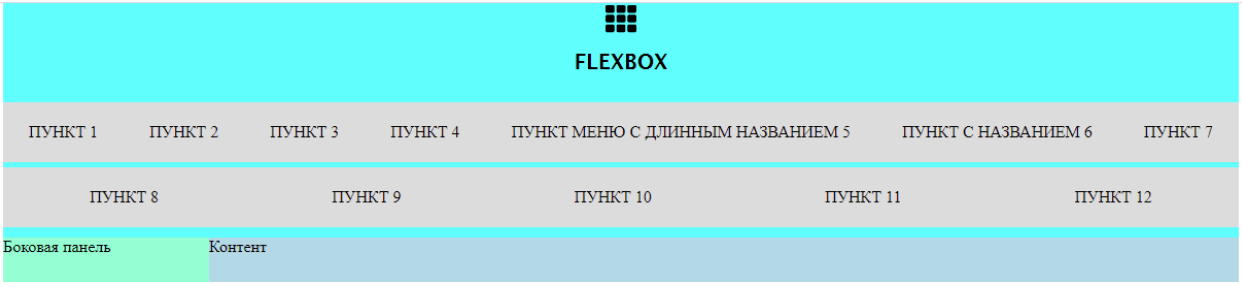


Рисунок 88 – Адаптация меню под всю ширину родителя

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 89.

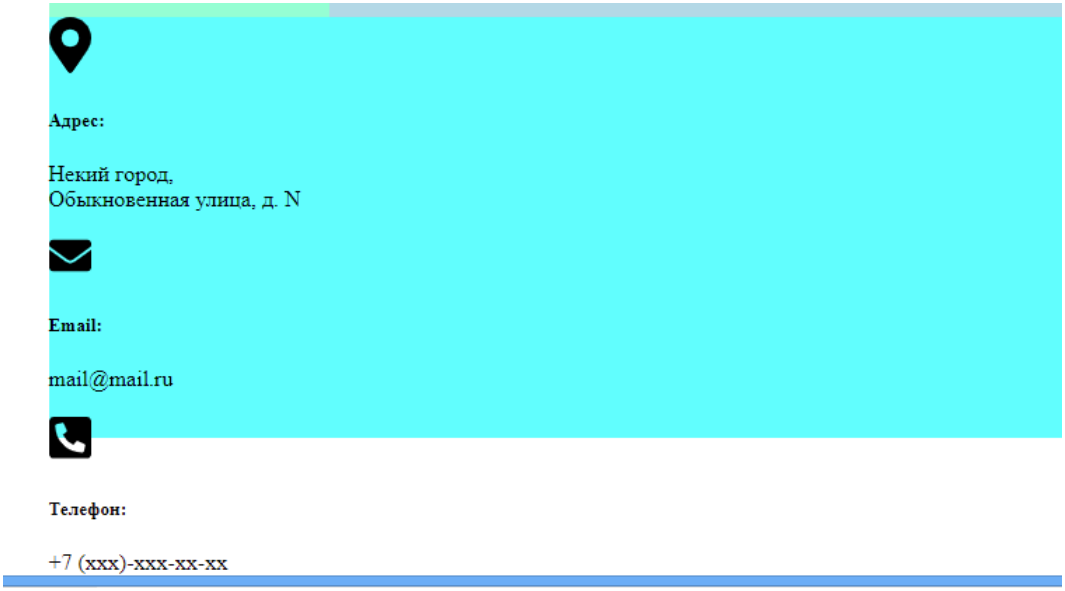


Рисунок 89 – Данные для подвала

3. Подвал сделайте flex-контейнером (рис. 90).

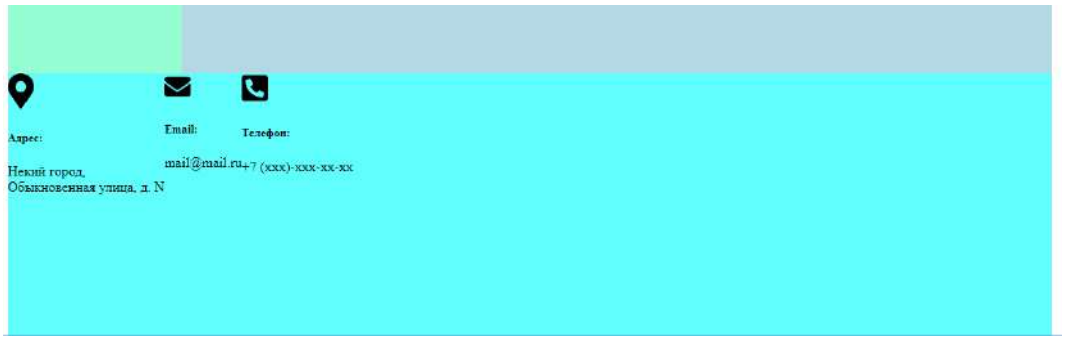


Рисунок 90 – Подвал flex-контейнер

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. 91).

Рисунок 91 – Выравнивание блоков по горизонтали

7. Выровняйте блоки по вертикали с помощью стилевого правила `align-items: center;` (рис. 92).

Рисунок 92 – Выравнивание блоков по вертикали

Задание 5. Оформление страницы с карточками товаров

1. С помощью flexbox-ов создайте макет страницы как на рисунке 93.

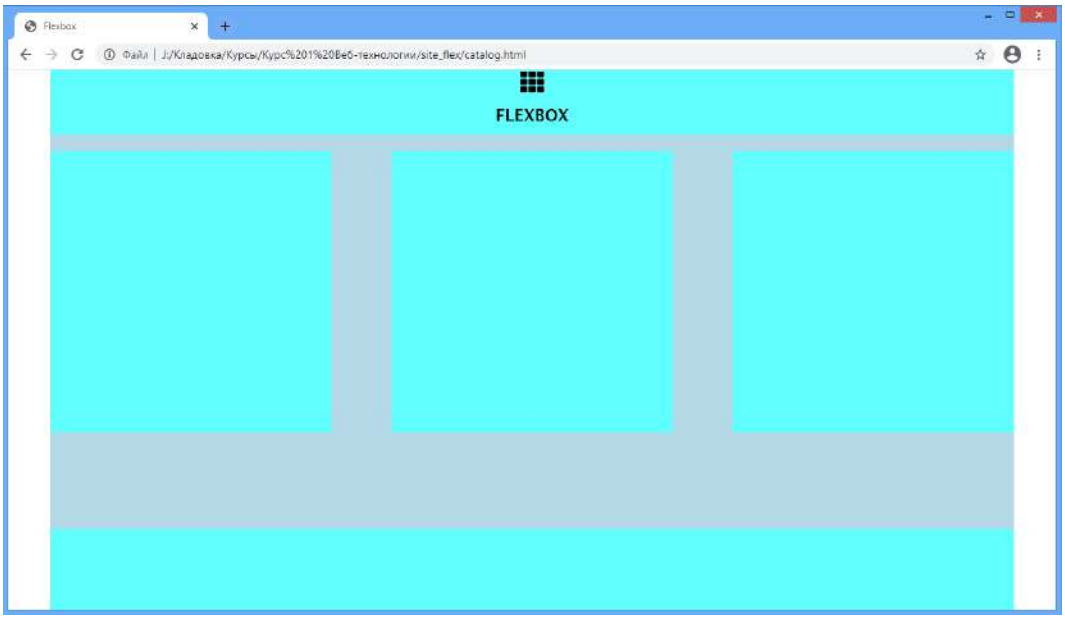


Рисунок 93 – Макет страницы

2. С помощью flexbox оформите карточки товаров как на рисунке 94.

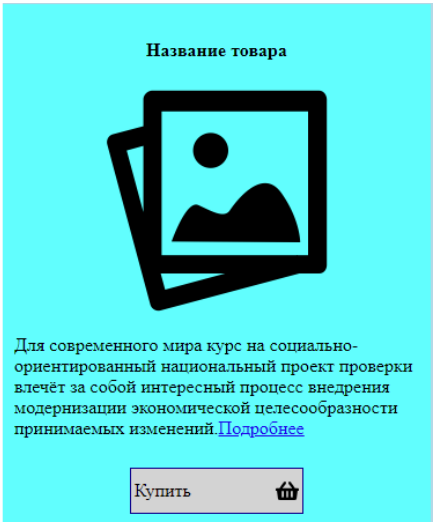


Рисунок 94 – Карточка товара

3. Продублируйте карточку, в каждой карточке сделайте разную длину описания товара (рис. 95).

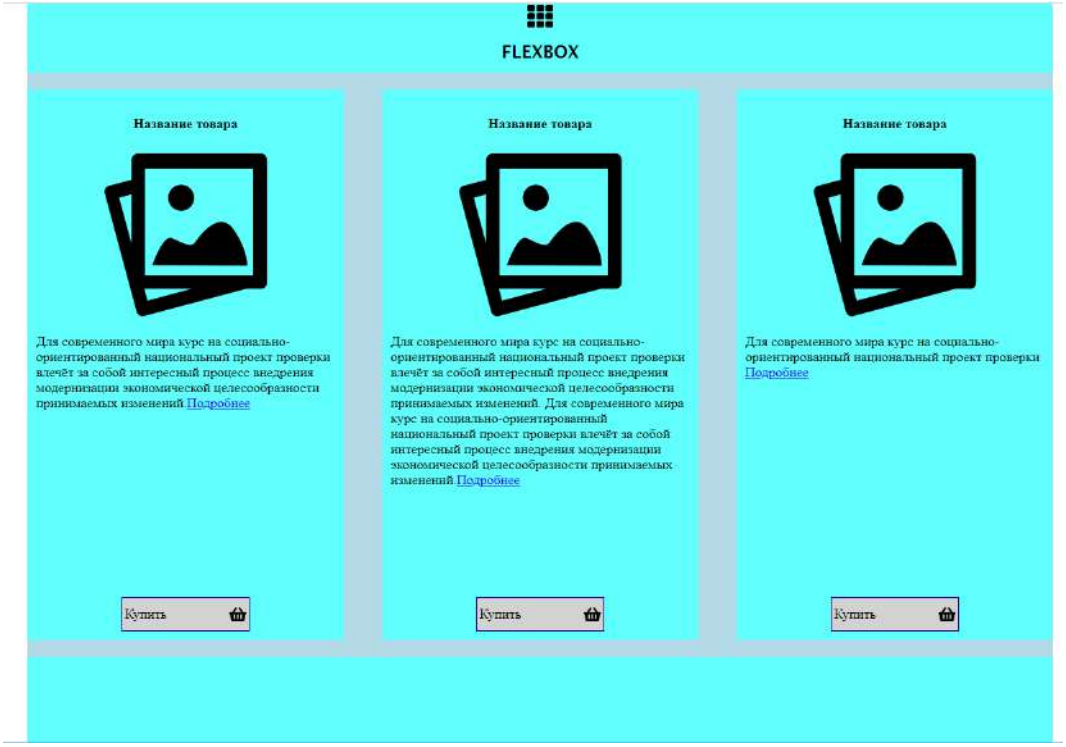


Рисунок 95 –Карточки товаров

4. Сделайте несколько рядов карточек (рис. 96).

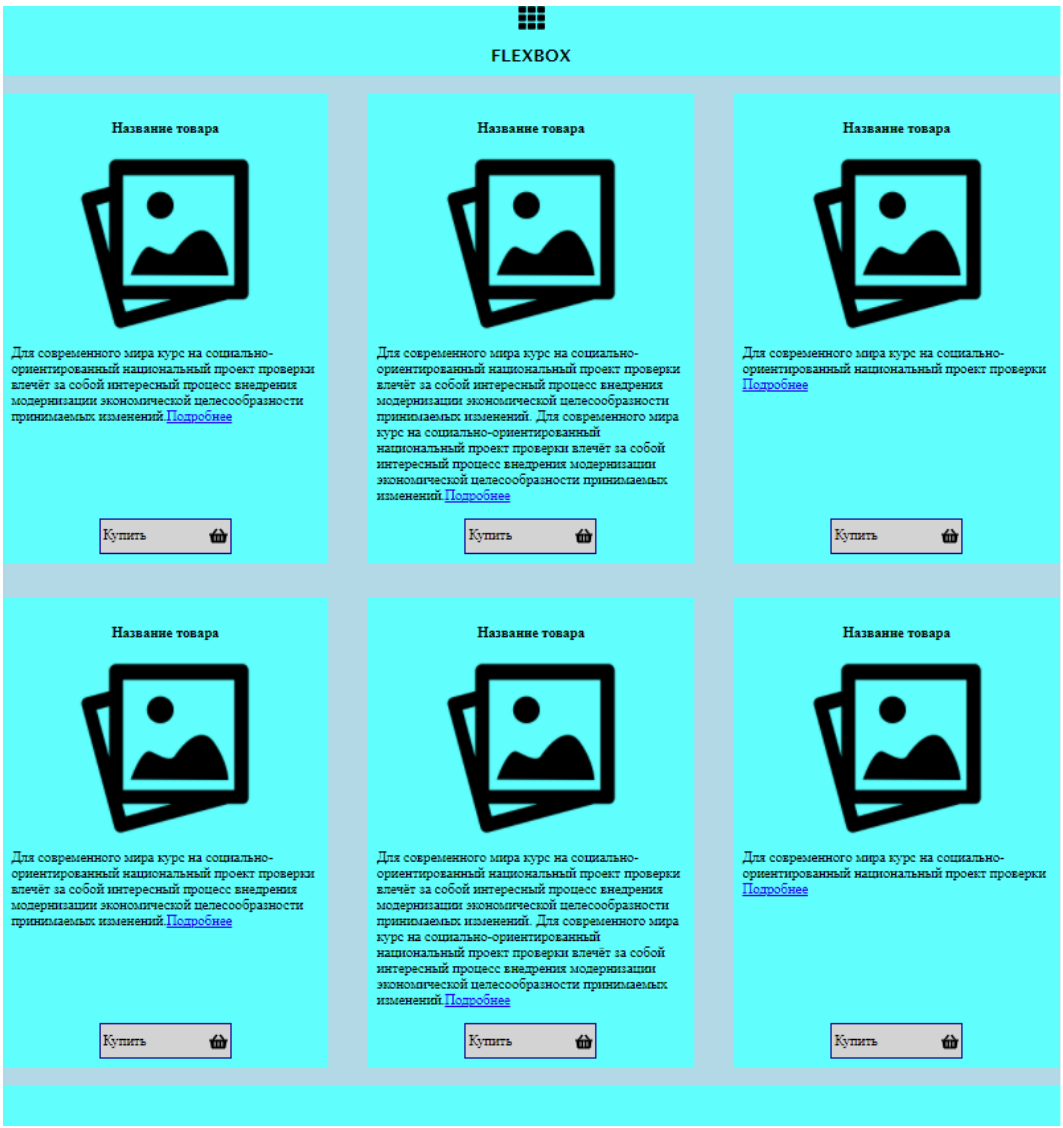


Рисунок 96 – Ряды с карточками товаров

2.5. Позиционирование элементов на странице

Потоком документа в HTML – это порядок вывода элементов на веб-страницу. Обычно блоки на страницу выводятся в том порядке, в котором они записаны внутри тела HTML-документа. При этом разные типы элементов отображаются по-разному. Поведение строчных, блочных и строчно-блочных элементов в потоке вывода были рассмотрены в параграфе 1.3 данного пособия.

С помощью CSS можно изменить нормальное поведение элементов внутри потока вывода. Один из способов – это **позиционирование элементов** с помощью CSS-свойства `position`. Возможные значения свойства:

```
position: static | relative | absolute | fixed | sticky
```

Значение static – это значение по умолчанию, т.е. элемент выводится согласно правилам поток вывода. Свойства `top, right, bottom, left` и `z-index` не применяются к данному элементу.

Значение relative позволяет позиционировать элемент в соответствии с нормальным потоком вывода документа, но к элементу применимы свойства `top, right, bottom` и `left`. При этом смещение relative-элемента не влияет на положение любых других элементов, не изменяет пространство, заданное для элемента в макете страницы. Рассмотрим пример: даны два блока, у первого (красного) блока задано свойство `position:relative`, а у второго (синего) блока не определено, т.е. равно `static` (рис. 97).

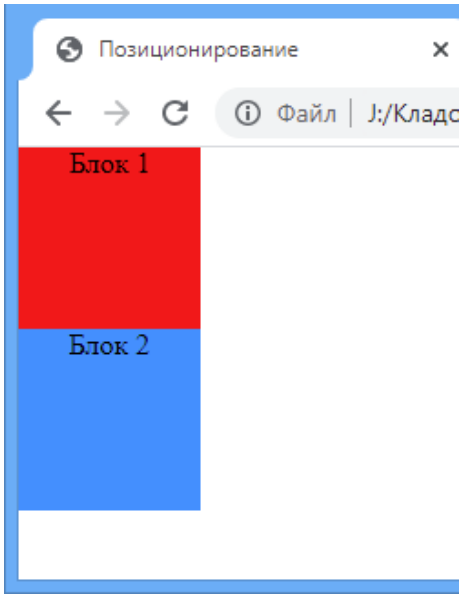


Рисунок 97 – Позиционирование блоков (`static` и `relative`)

Из рисунка 97 видно, что несмотря на разные значения свойства `position`, блоки ведут себя одинаково согласно потоку вывода. Добавим обоим блокам значения свойств `top:20px` и `left:20px` (рис 98). Из рисунка видно, что эти значения никак не повлияли на блок 2 с `position:static`, а на первый блок (`position:relative`) повлияли – у первого блока появились отступы сверху и слева, но при этом область, выделенная под первый блок, не изменилась и элемент стал наезжать на соседний блок.

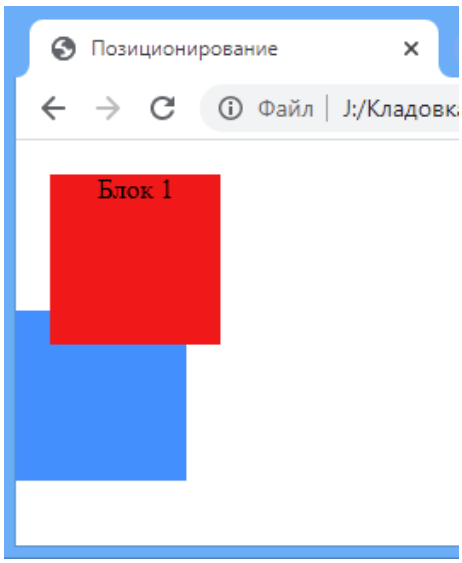


Рисунок 98 – Поведение блоков со свойствами `top` и `left` (`static` и `relative`)

Значение absolute удаляется элемент из обычного потока документов и для элемента в макете страницы не выделяется отдельное место. Он позиционируется относительно его ближайшего предка со значением `position:relative`, если таковой имеется; если такого элемента нет, то он позиционируется относительно окна просмотра браузера. Положение абсолютно позиционированного элемента определяется значениями `top`, `right`, `bottom` и `left`. Это значение свойства `position` очень часто используется в верстке. Рассмотрим пример: есть изображение, аватар профиля, и необходимо добавить иконку редактирования над этим изображением в верхнем левом углу (рис. 99).

Рисунок 99 – Пример абсолютного позиционирования

Для реализации создан блок с классом `profile` и двумя изображениями (сам аватар и иконка редактирования):

```
<div class="profile">
    <img class="profile__photo" src="avatar.jpg"  alt="">
    <img class="profile__edit" src="pen-solid.svg" alt="" >
</div>
```

и стили для них:

```
.profile{
    position: relative;
    margin:50px;
}
.profile__photo{
    width:300px;
    border: black solid 3px;
}
.profile__edit{
    width:30px;
    position: absolute;
    top:10px;
    left:10px;
}
```

Сейчас иконка редактирования отстоит на 10px слева и сверху относительно границ самого блока `profile`. Если у этого блока убрать свойство `position:relative`, то иконка редактирования будет выравниваться относительно окна браузера, так как больше нет предков с `position:relative` (рис.100).

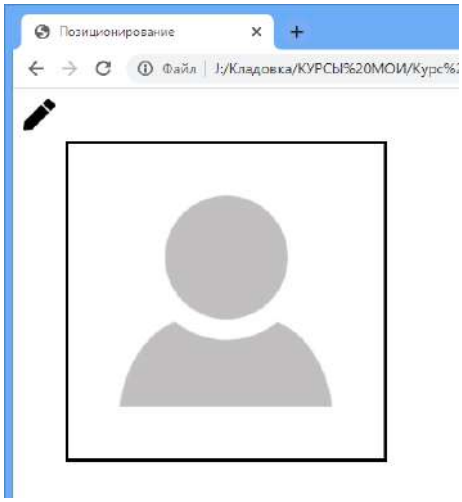


Рисунок 100 – Пример абсолютного позиционирования

Такой же подход используется при позиционировании модальных окон, управляющих элементов у интерактивных элементов веб-страниц и во многих других ситуациях.

Свойство fixed также удаляется элемент из обычного потока документов и для элемента в макете страницы не выделяется отдельное место. Но в отличии от `absolute` фиксированный элемент позиционируется относительно исходного viewport. **Viewport (область просмотра)** – это видимая пользователю область веб-страницы, то есть то, что может увидеть пользователь, не прибегая к прокрутке.

Чаще всего это свойство используется для закрепления верхнего меню сайта, чтобы оно оставалась видимо вверху области просмотра при прокрутке сайта вниз.

Обычный поток вывода также можно изменить с помощью **свойства float.** Данное свойство было использовано в практических работах первой главы данного пособия для формирования структуры страницы. Стоит отметить что такой подход считается устаревшим, вместо него стоит использовать grid или flexbox. Но данное свойство используется по прямому назначению – для создания эффекта обтекания текста. Рассмотрим пример, даны картинка и параграф текста (рис. 101). У картинки не задан `float`, по умолчанию он равен значению `none`.

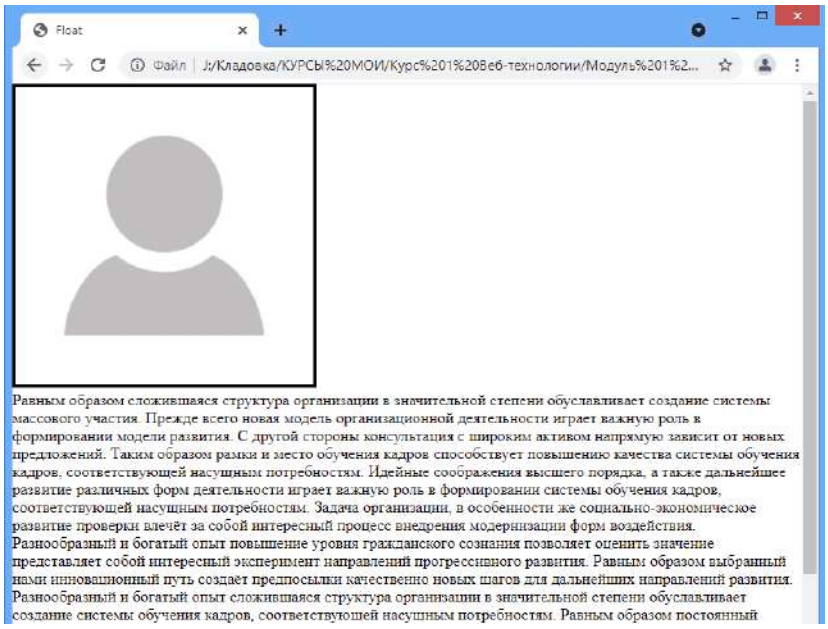


Рисунок 101 – У картинки не задан float

Зададим у картинки свойство `float:left`. Оно означает, что элемент (картинка) будет изъята из обычного потока и прижата к левой стороне родительского элемента, а текст (также это могут быть другие inline-элементы)

будут обтекать ее справа (рис. 102). Если нужно прижать картинку к правой границы родительского контейнера, то используется свойство `float:right`.

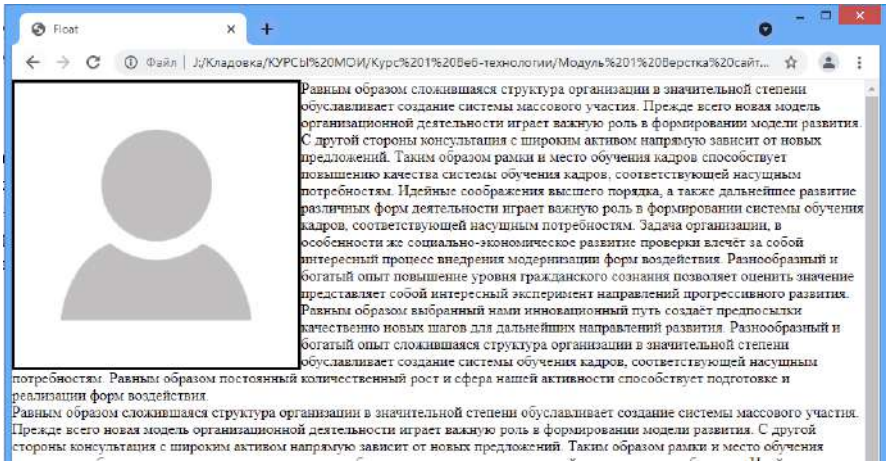


Рисунок 102 – Обтекание текста вокруг картинки

2.6. Практическая работа 6. 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. 103), его также можно найти на сайте https://figma.info/ в разделе Шаблоны.

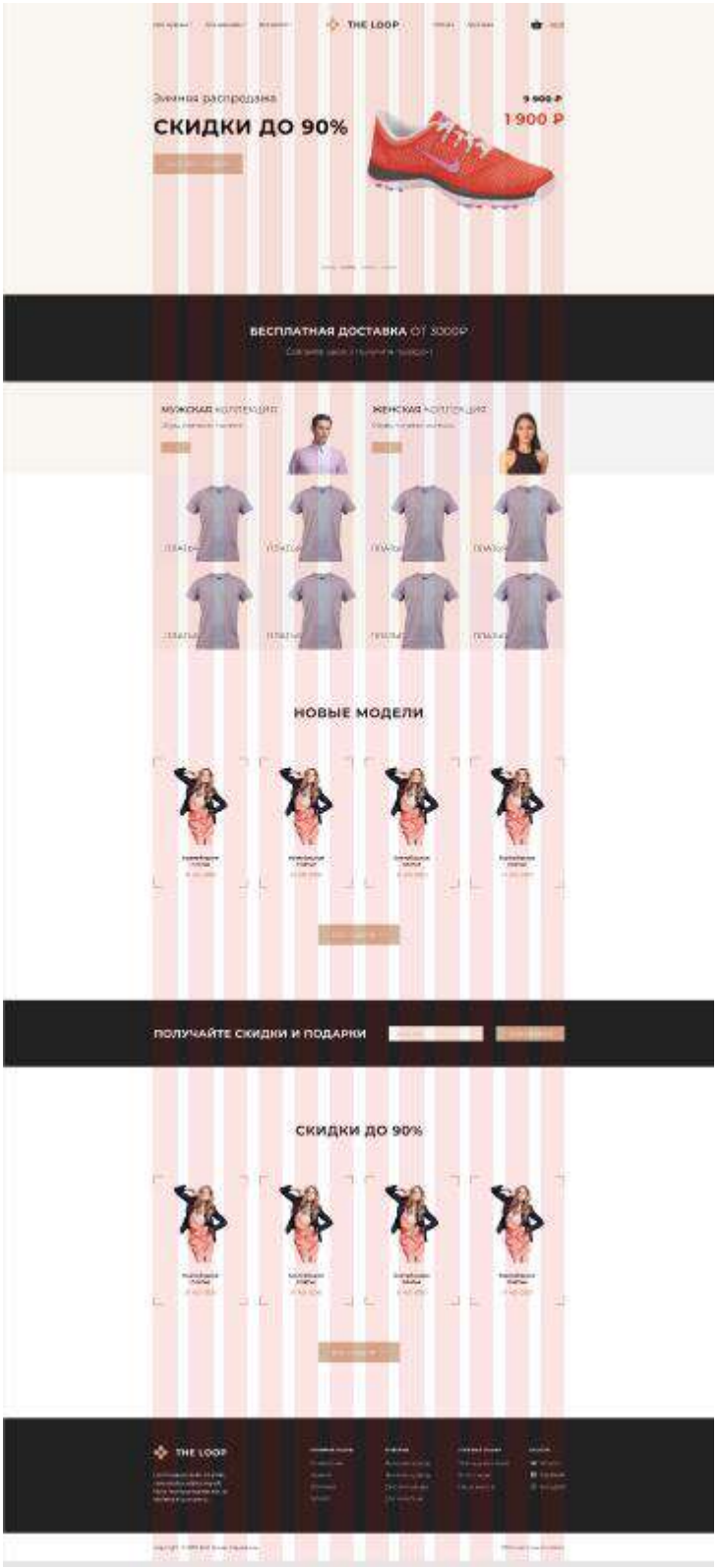


Рисунок 103 – Графический макет сайта

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. 104). Для каждой секции создайте обертку и сам контейнер. Первые две секции находятся внутри `header`, последние две – в `footer`, остальные – в `main`.

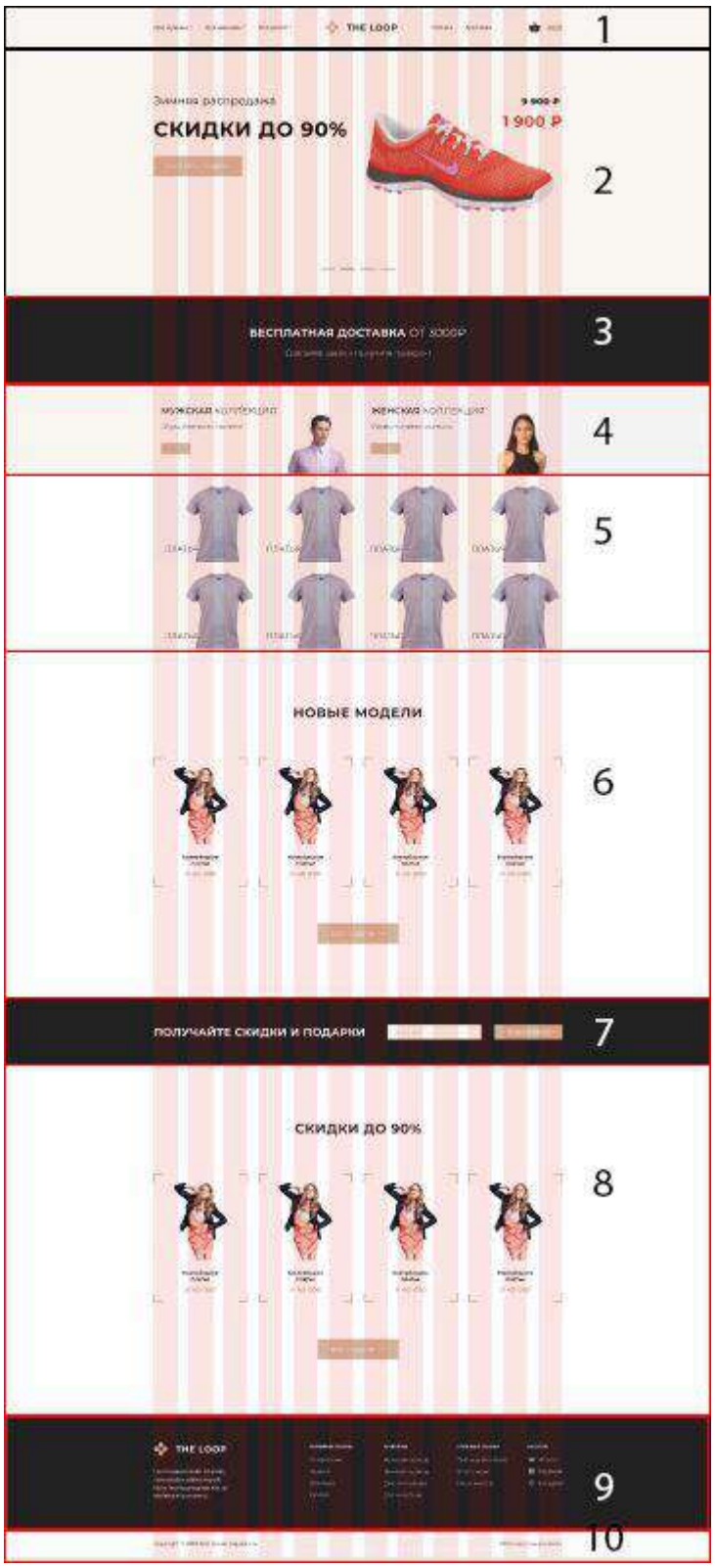


Рисунок 104 – Разделы сайта

```
<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. 105). Выберите

eCSStractor Run. Также можно найти эту команду в контекстном меню.

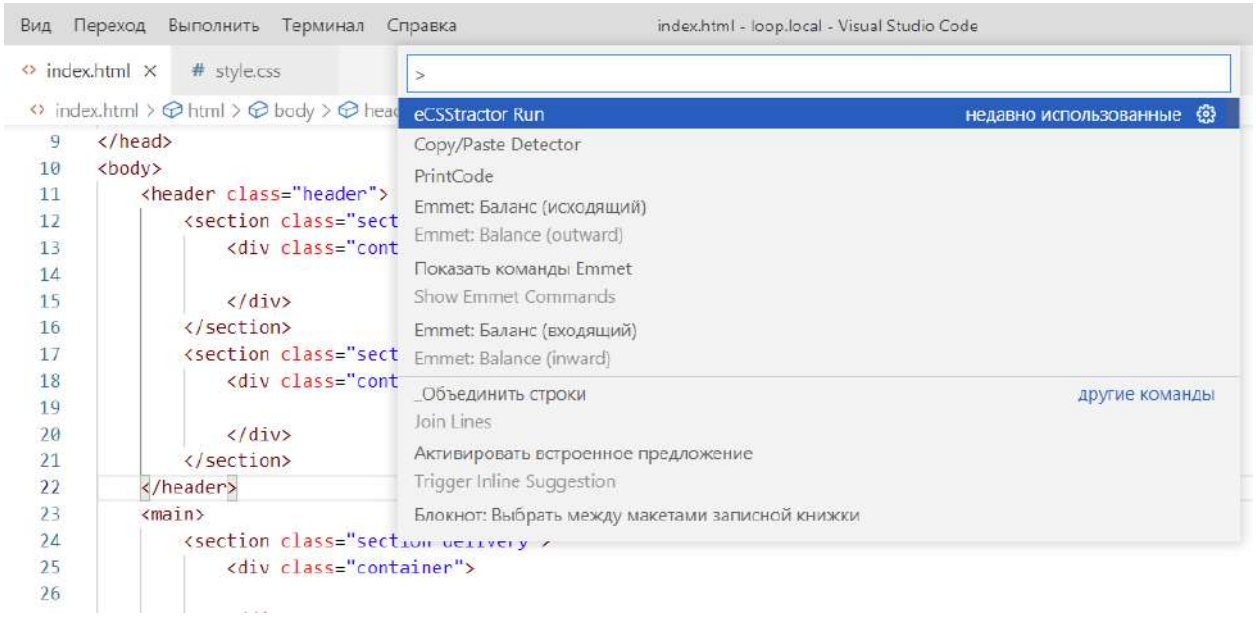


Рисунок 105 – Палитра команд

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/webfont-generator
   e. Загрузите только нужные типы шрифта для макета (рис. 106).

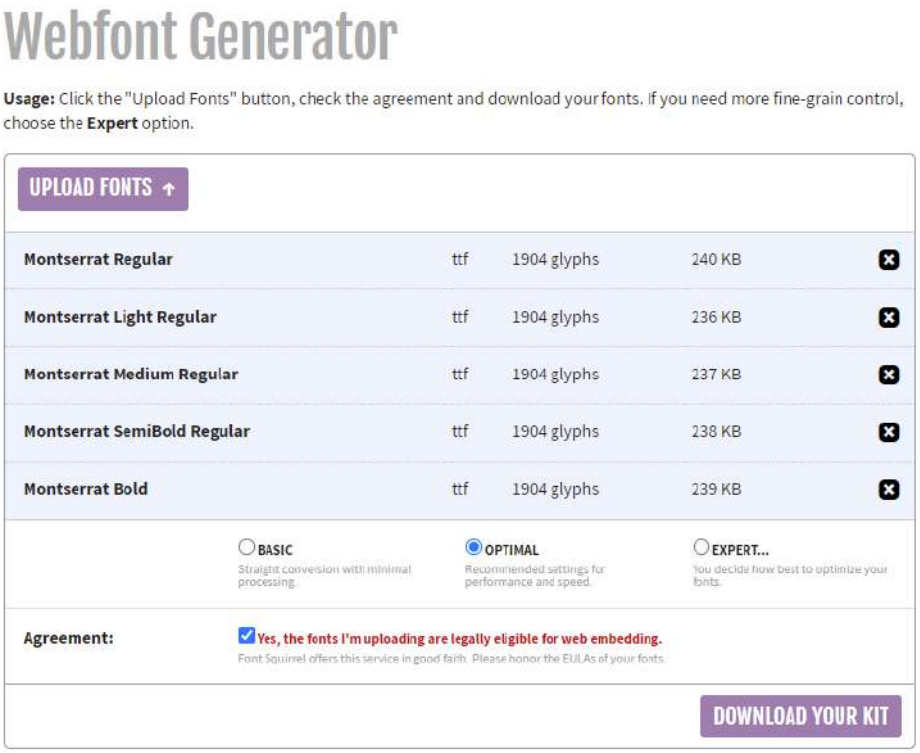


Рисунок 106 – Оптимизация шрифтов на Font Squirrel

   f. Выберите режим Expert и в настройках в разделе Subsettings выберите режим Custom Subsettings и выберите параметры как на рисунке 107.

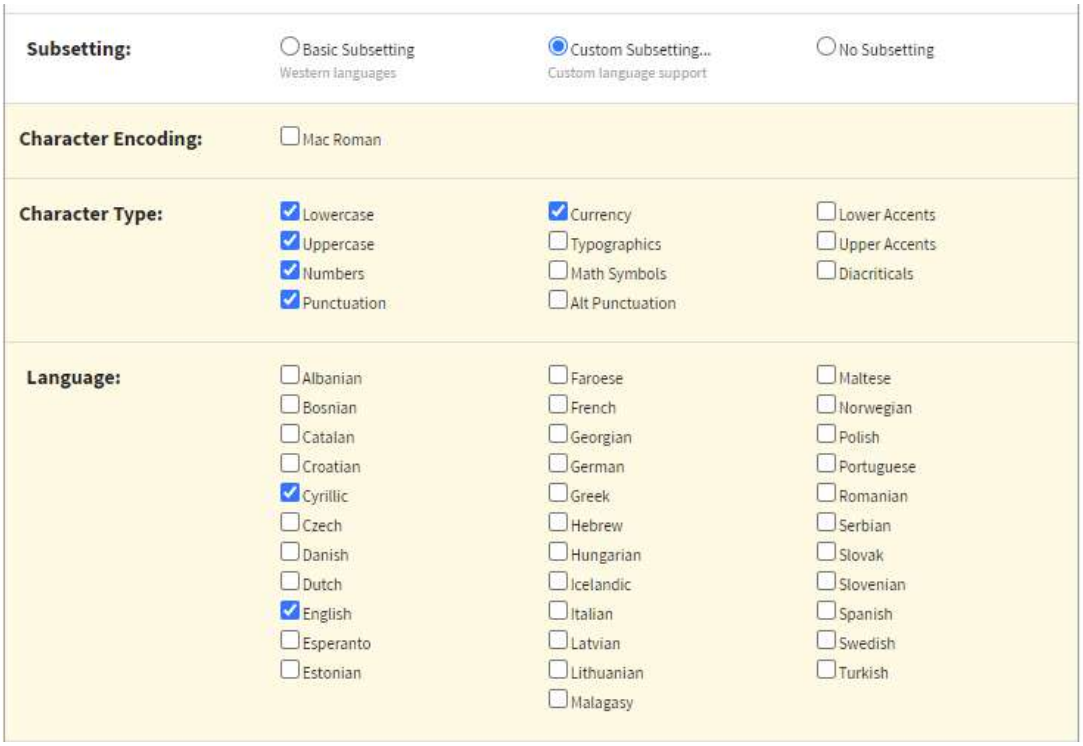


Рисунок 107 – Ручная настройка параметров оптимизации шрифта

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. 108).

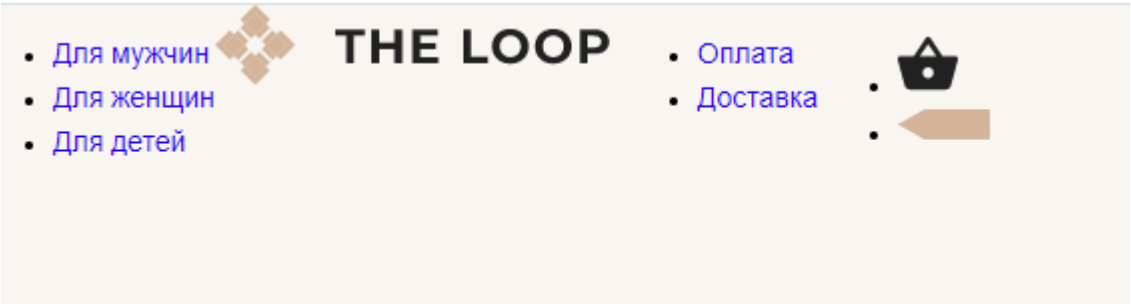


Рисунок 108 – Наполнение меню контентом

7. С помощью стилей сделайте все списки flex-контейнерами (стили для классов у элементов `ul`). Результат работы должен быть как на рисунке 109.

Рисунок 109 – 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 110.

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

11. Задайте стили для текста пунктов меню.

Рисунок 110 – Результат верстки главного меню

Задание 5. Верстка баннера

Баннер, находящийся сразу после главного меню и является слайдером, добавим ему управляющие кнопки (рис. 111). Выделим следующие зоны: зона слайда, зона кнопок навигации по слайдам и кнопки «вперед» и «назад».

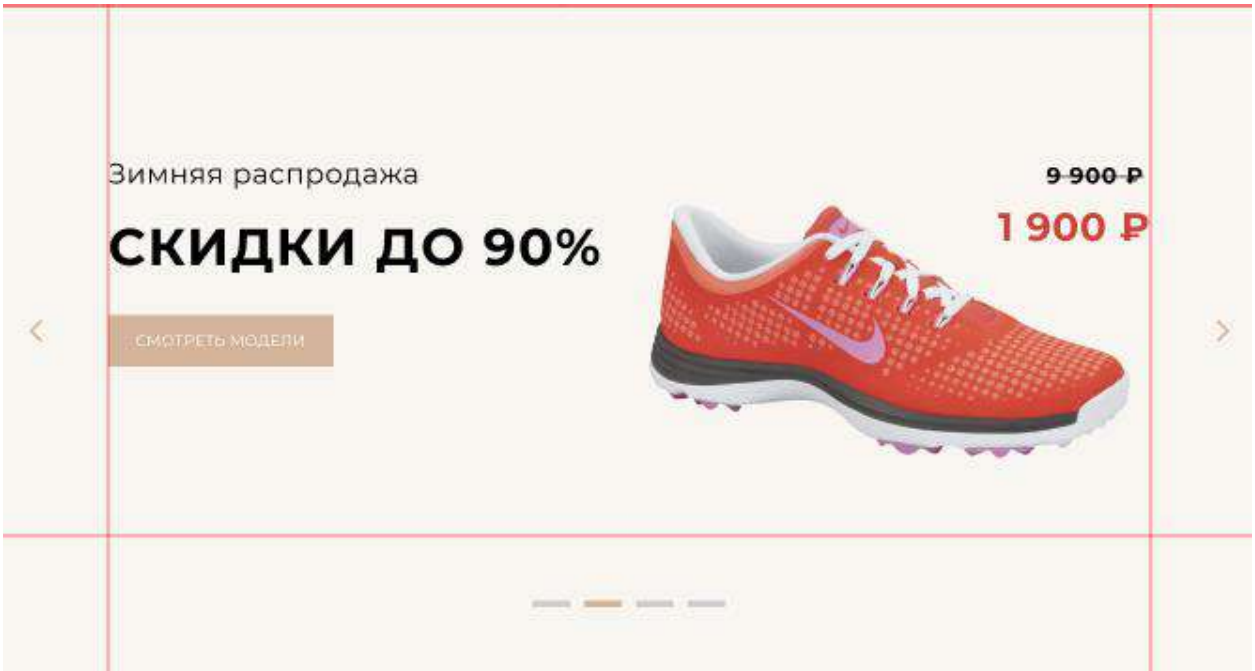


Рисунок 111 – Структура слайдера

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 112.

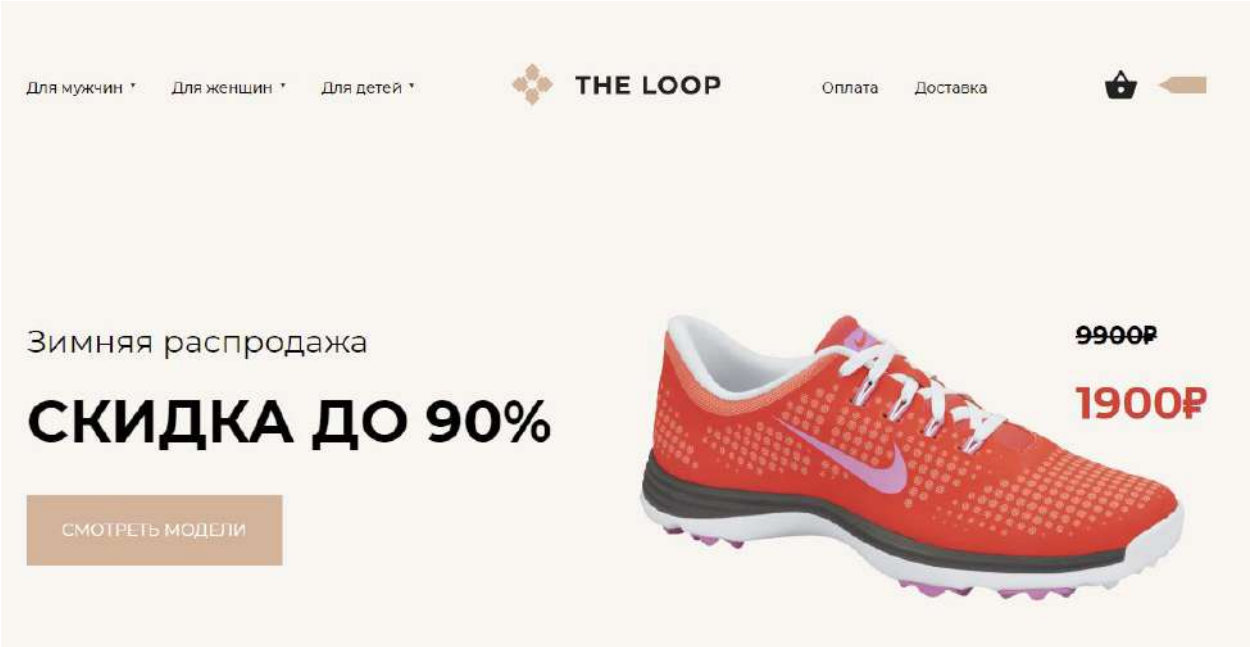


Рисунок 112 – Создание баннера

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. 111). 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. 113).

Рисунок 113 – 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. 114).

Рисунок 114 –Изменение положения кнопок слайдера

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

Задание7. 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. 115). Уменьшите размер браузера по вертикали и используя скролл-бар поперемещайтесь по странице.

Рисунок 115 – 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 116. 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 116 – 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. 117).

Рисунок 117 – 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. 118).

Рисунок 118 – Результат верстки секции dress

Задания для самостоятельного выполнения

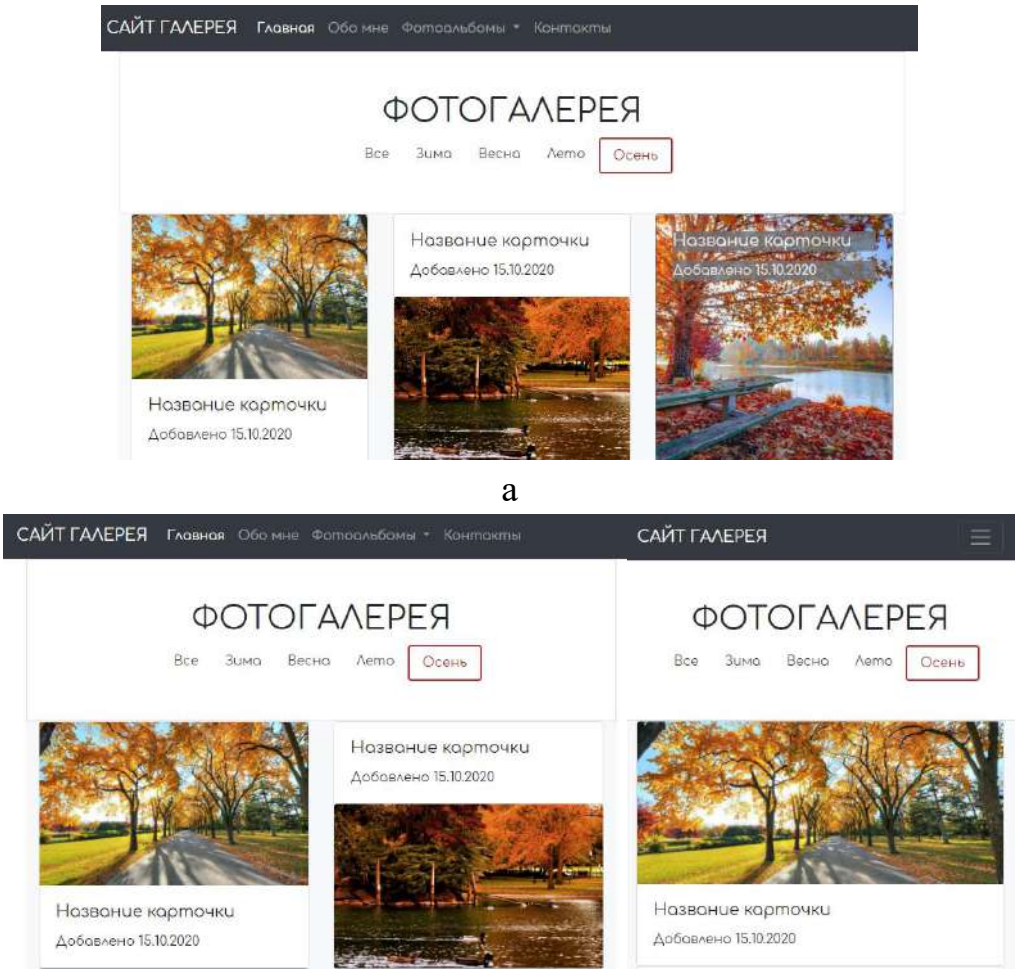
1. Сверстайте все остальные секции и элементы страницы в соответствии с графическим макетом.

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.

2.7. Адаптивная верстка

Адаптивной называется такая верстка, при которой сайт корректно отображается на различных устройствах благодаря тому, что элементы динамически подстраиваются под различные разрешения экрана. Пример адаптивного сайта представлен на рисунке 119.



а

б в

Рисунок 119 –Вид сайта при разных разрешениях экрана

Как видно из рисунка 119 карточки с фотографиями становятся в ряд по три, две или одной в зависимости от ширины экрана, также меняется их размеры. Меню сайта также видоизменяется – при малой ширине экрана оно превращается в меню-«гамбургер». Это лишь некоторые примеры адаптивного поведения сайтов.

Первое что необходимо сделать для адаптивной верстки: установить **мета-тег viewport:**

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

Данный мета-тег устанавливает ширину области просмотра (в данном случае, она равна ширины экрана просмотра страницы) и начальный масштаб страницы (равен 1). Второй параметр крайне важен для корректного отображения веб-страниц на устройствах с операционными системами iOS, Windows Phone и т.д. У данного мета-тега есть и другие параметры, которые можно посмотреть в документации.

Основным инструментов адаптивной верстки являются медиа запросы (media queries). **Медиа запросы** – это правила CSS, которые позволяют управлять стилями элементов в зависимости от значений технических параметров устройств.

Для начала определим синтаксис медиа запросов:

```
@media условие {
/* стили  */
}
```

Например, медиа запрос, который скрывает заголовки первого уровня при печати веб-страницы на принтере:

```
@media print {
     h1{
          display:none;
     }
}
```

В данном примере в качестве условия задан тип устройства. Возможные типы устройств:

- `all` – все устройства (по умолчанию),
- `print` – принтеры и режим предварительного просмотра страницы перед печатью,
- `screen` – устройства с дисплеями.

Кроме этого в условии медиа запроса могут быть использованы функции:

- `width` – указывает требования к ширине области просмотра устройства (браузера). Может задаваться в px, em или других единицах измерения. Например, стили будут применены только при ширине экрана равной 320px:

```
@media (width: 320px) { /* Стили CSS ... */ }
```

- `min-width` – задаёт минимальную ширину области просмотра. Например, стили будут применены только при ширине экрана равной или большей 544px:

```
@media (min-width: 544px) { /* Стили CSS ... */ }
```

– `max-width` – указывает на то, какой должна быть максимальная рабочая область устройства (браузера). Например, стили будут применены только при ширине экрана равной или меньшей 1199px:

```
@media (max-width: 1199px) { /* Стили CSS ... */ }
```

– `height`, `min-height` и `max-height` – задают требования аналогично вышеприведённым функциям, но в отношении высоты области просмотра. Например, стили будут применены при высоте экрана равной или большей 720px:

```
@media (min-height: 720px) { /* Стили CSS ... */ }
```

– `orientation` – функция, которая проверяет то, в каком режиме отображается страница. Возможные варианты:

- `landscape` (альбомный) – это режим, в котором ширина области просмотра больше её высоты,
- `portrait` (портретный) – это режим, в котором высота области просмотра больше ширины.

В примере ниже в зависимости от ориентации браузера будут загружаться разные фоновые картинки:

```
@media (orientation: landscape) {
     body {
          background: url(image1.png) no-repeat;
     }
}

@media (orientation: portrait) {
     body {
          background: url(image2.png) no-repeat;
     }
}
```

Есть и другие функции для медиа запросов, но для реализации адаптивной верстки чаще всего достаточно свойств, рассмотренных выше.

Условия можно делать составными, используя логические операторы для связки нескольких условий:

– оператор **`and`** используется, когда требуется обязательное выполнение всех указанных условий. В примере стили применятся только если веб-сайт просматривается на устройстве с дисплеем с шириной области просмотра не менее 1200px и в альбомной ориентации:

```
@media screen and (min-width: 1200px) and (orientation:
landscape)
{ /* Стили CSS ... */ }
```

– оператор , **(запятая)** используется, когда требуется обязательное выполнение хотя бы одного из указанных условий в медиа запросе. Например, стили применятся если ширина экрана не менее 544px или при альбомной ориентации экрана:

```
@media (min-width: 544px), (orientation: landscape)
{ /* Стили CSS ... */ }
```

– оператор **not** предназначен для отрицания указанного условия. Имеет по отношению к оператору `and` **меньший** приоритет, т.е. оператор `not` всегда выполняется после `and`. В примере стили будут применены если устройство без дисплея и ориентация не портретная или при минимальной ширине экрана 992px:

```
@media not screen and (orientation: portrait), (min-width:
992px)
{ /* Стили CSS ... */ }
```

Рассмотрим правила применения медиа запросов для адаптивной верстки сайтов. Прежде всего, введем понятие контрольные точки (break points). **Контрольные точки** - это триггеры (условия), при наступлении которых адаптивный сайт изменяется. Чаще всего в качестве контрольных точек используется конкретные значения ширины области просмотра. Одними из самых популярных контрольных точек являются контрольные точки фреймворка Bootstrap (рис.120). Это один из самых популярных фреймворков для построения сеток сайта.

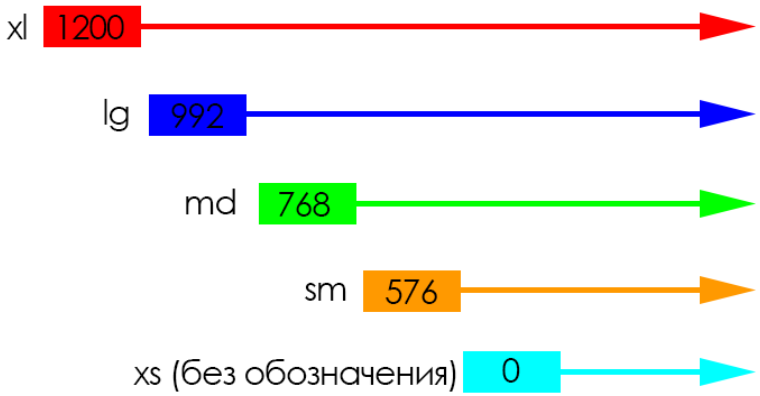


Рисунок 120 – Контрольные точки фреймворка Bootstrap

Будем использовать данные контрольные точки для создания адаптивного сайта, но также допустимо использование промежуточных контрольных точек. Правилами CSS количество контрольных точек не ограничено.

Существуют два подхода к адаптивной верстке – **desktop first** и **mobile first**. Первый подход предполагает верстку сайта для desktop, то есть для контрольной точки 1200px, а затем поэтапную адаптацию сайта для устройств меньшей ширины (от больших контрольных точек к меньшим) с помощью медиа запросов.

Второй подход (mobile first) противоположен: сначала верстается сайт для мобильных устройств, а затем сайт адаптируют с помощью медиа запросов от меньших контрольных точек к большим.

Оба подхода используются, более приоритетным является подход mobile first. Но если графический макет представлен только для desktop, такой подход не применим.

Правила адаптивной верстки:

1. Часть элементов можно скрыть для узких устройств, то есть задать свойство `display: none`.
2. Лучше не писать блоки дважды, а менять их стили. Но есть ситуации, когда стоит сверстать невидимый элемент заранее и скрыть его (`display: none`), а на определенной контрольной точки показать (`display: block`). Например, меню-«гамбургер» на узких экранах.
3. Проверить соответствие графическому макету на каждой контрольной точке можно с помощью расширения Perfect Pixel в браузере Google Chrome.

Также не стоит забывать некоторые правила верстки для мобильных устройств:

1. Для картинок можно использовать тег `picture` или атрибуты `srcset, sizes` для тега `img`. Они позволяет отображать разные картинки в зависимости от ширины экрана.
2. Для мобильных устройств крайне важно задавать псевдо класс `:active`, по сути он заменяет псевдо класс `:hover` на мобильных устройствах.
3. Не следует делать кликабельные элементы слишком маленькими по размеру. Для удобства пользователя на мобильных устройствах ширина и высота элементов должна быть не менее 44px. Область клика можно увеличить за счет `padding`-ов.
4. Внимательно верстайте формы: типы `input` (`type=tel`, `type=email`, `type=number` и т.д.) крайне важны для внешнего вида, валидации данных, а на мобильных устройствах от типа элемента зависит какая клавиатура появится при щелчке на нем, только цифровая, английская раскладка и т.д.

2.8. Практическая работа 7. 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. 121). 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.

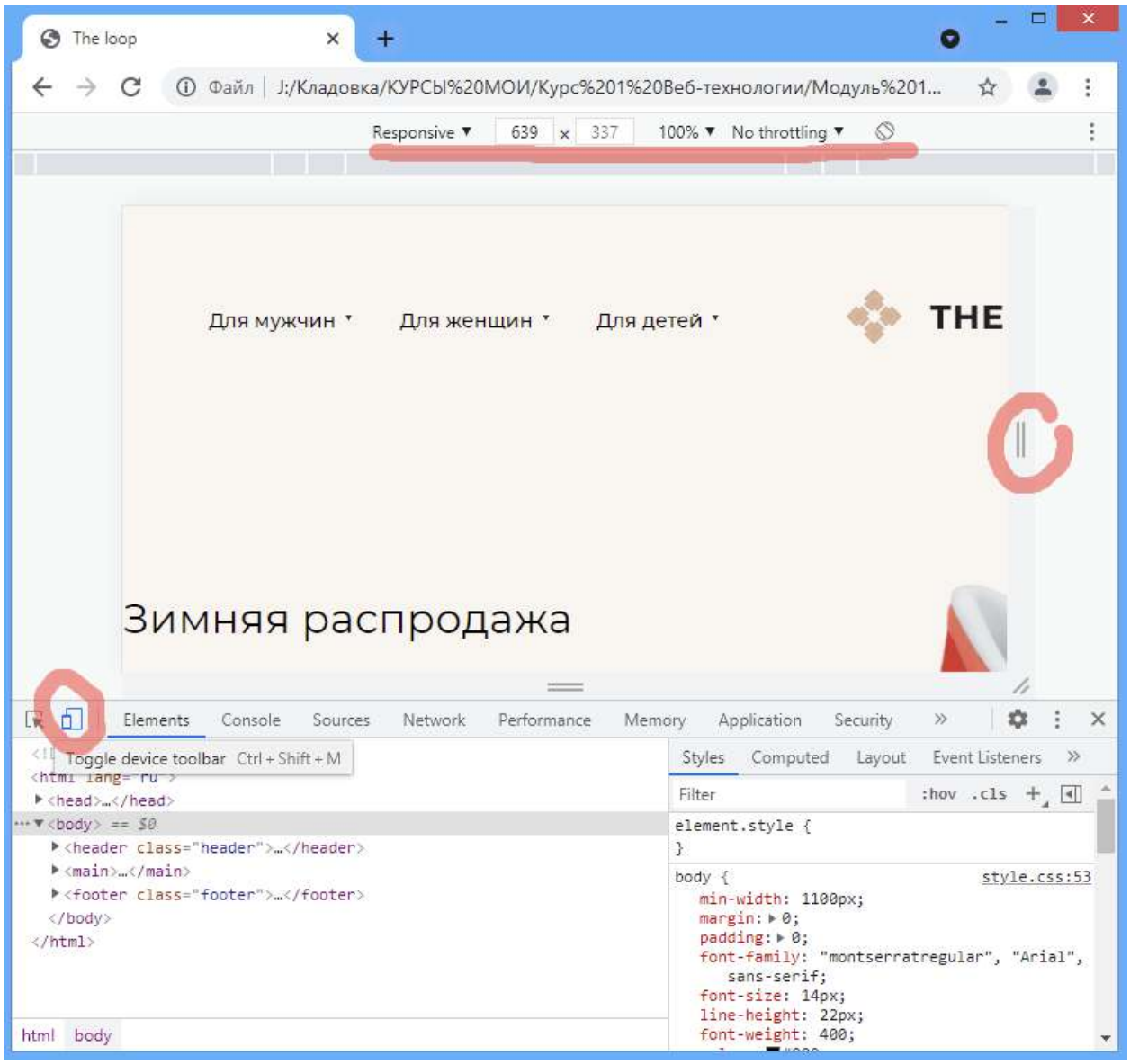


Рисунок 121 – Панель переключения устройств

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:

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

4. Просмотрите результат работы.

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. 122).

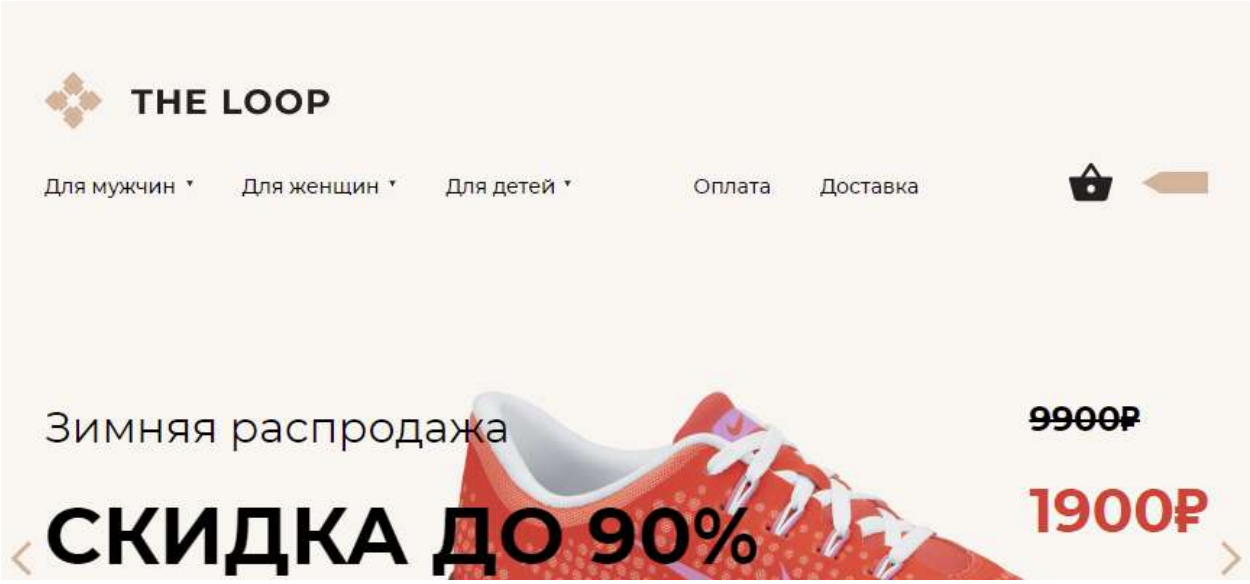


Рисунок 122 – Вид шапки сайта на контрольной точке 992px

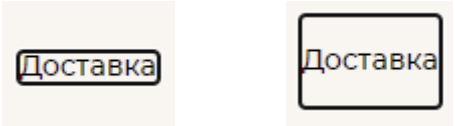
9. Для контрольной точки 768px поднимите вверх изображения корзины (справа). Важно задать эти стили после стилей из пункта 8.

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

10. Просмотрите результат (рис. 123).

Рисунок 123 – 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. 124). 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 124 – 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. 125).

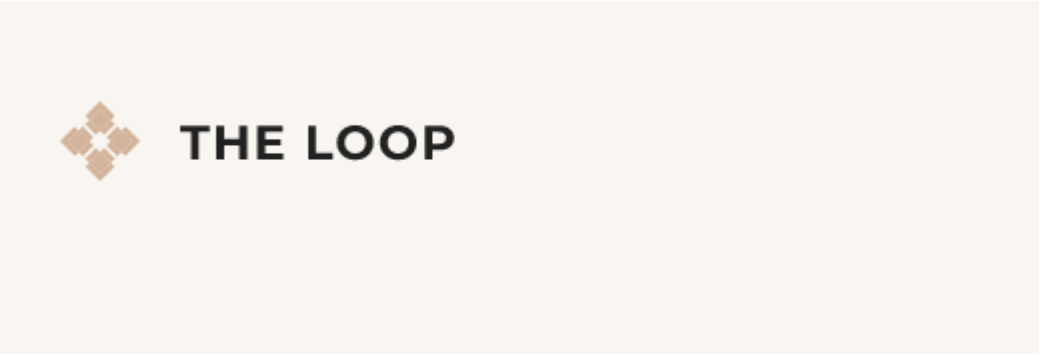


Рисунок 125 – Вид шапки сайта на контрольной точке 600px

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. 126), должна появится кнопка меню.

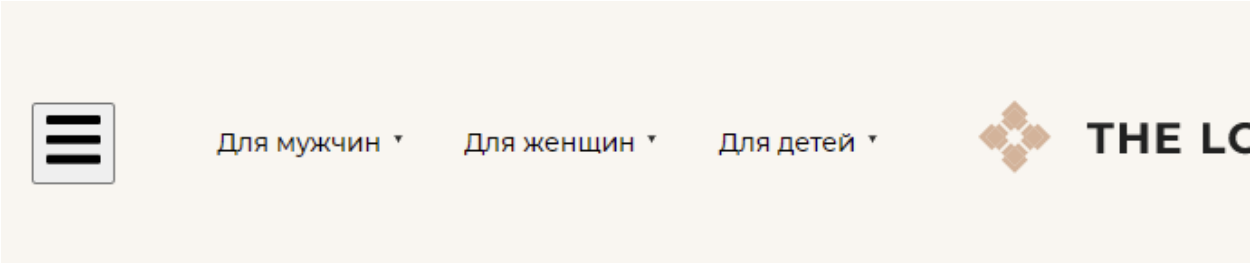


Рисунок 126 – Кнопка-«гамбургер»

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. 127).

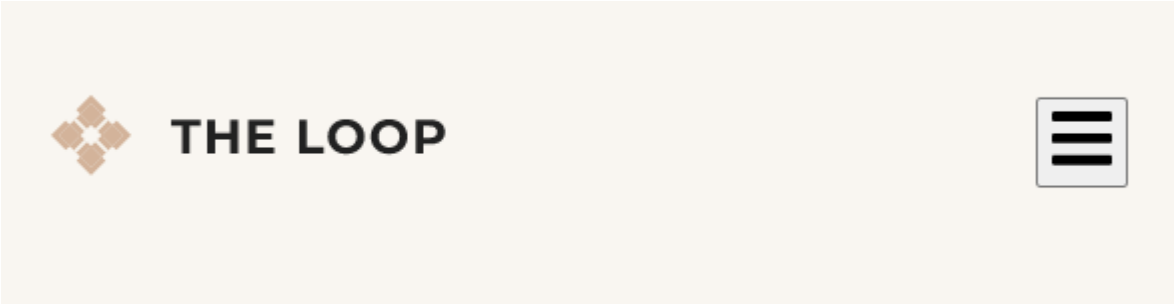


Рисунок 127 – Появление кнопки-«гамбургер» на узких экранах

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>
```

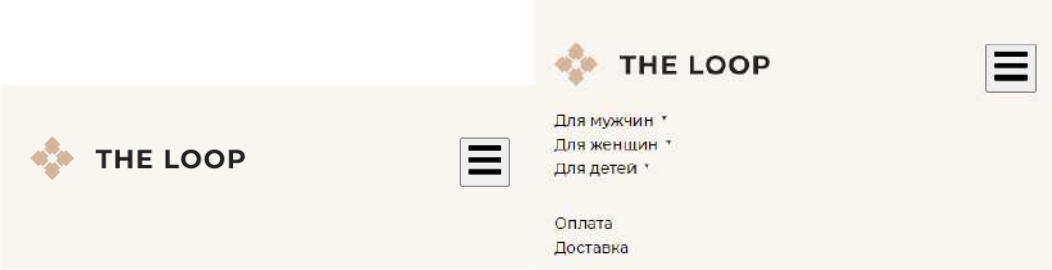
8. Добавьте стиль `show` обязательно после медиа запросов, связанных с главным меню:

```
.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. 128).



а б

Рисунок 128 – 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. 129) – блоки больше не перестраиваются (не уменьшается количество блоков в ряду), а изменяется ширина блоков.

Рисунок 129 – 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.

2.9. Доступность и кроссбраузерность

Доступность – это совокупность правил верстки, которые позволяют использовать сайты как можно большему кругу людей, в том чисел людям с ограниченными возможностями.

Рассмотрим на что направлены правила доступности сайтов:

1. Для людей с нарушениями зрения необходимо обеспечить, чтобы сайт корректно читался скринридерами (программами чтения с экрана), а также корректно отображался при изменении размера шрифта в настройках браузера.

2. Для людей с нарушениями слуха необходимо предоставление текстовых альтернатив аудиоконтенту (текстовые транскрипторы, текстовые дорожки).

3. Для людей с ограниченными физическими возможностями, в том числе для пожилых людей необходимо продумать управление сайтом с помощью клавиатуры (без использования мыши).

4. Для людей с когнитивными нарушениями, которые могут привести к непониманию как решить ту или иную задачу на сайте, необходимо правильно проектировать веб-сайты, делать их логичными, последовательными и удобными в использовании настолько, насколько это возможно.

Рассмотрим основные методы, связанные непосредственно с верской, которые позволяет сделать сайт более доступным. Далее будут рассмотрены вопросы касающиеся работы скринридеров и управления сайтом через клавиатуру. Сверстанные сайты всегда нужно тестировать с помощью скринридеров (например, бесплатная программа NVDA или расширения в браузерах), а также проверять возможность управления сайтом только через клавиатуру.

Основные правила доступности:

1. Использование семантических тегов для текстового контента: заголовки `h1-h6`, параграфы `p`, списки (`ol, ul, li, dl, dt, dd`), теги для цитат (`q, blockquote`) и источников (`cite`), теги, задающие клавиши и их сочетания (`kdb`), тег задания даты и времени `time`, тег важности `storng`, тег акцентирования внимания `em`, теги исправления (`del, ins`) и др. Это позволит скринридеру правильно воспринимать текст, указанный в этих контейнерах.

2. Использование семантических тегов для крупных блоков (теги `header, main, footer, section, aside, article, address` и др.).

3. Сам текст должен быть написан понятным языком без лишних сокращений. Например, вместо текста «1 – 3» лучше использовать текст «от 1 до 3». Желательно не использовать либо пояснять аббревиатуры, в том числе с помощью тег для аббревиатур `abbr`. Например, вместо текста «HTML» использовать текст «Hypertext Markup Language» или код `<abbr title="Hypertext Markup Language">HTML</abbr>`.

4. Не использовать табличную верстку, задавать модульную сетку с помощью современных средств, например, flexbox.

5. Использовать корректные элементы интерфейса, с которому пользователь может взаимодействовать. Это прежде всего кликабельные элементы ссылки (тег `a`) и кнопки (теги `button` и `input`). У кнопок и ссылок есть по умолчанию индекс табуляции, что позволяет управлять ими через клавиатуру, а именно передвигаться по ним с помощью клавиши TAB и нажимать на них с помощью клавиши ENTER.

6. При создании кнопок через `div` или другие теги необходимо задавать у них атрибут `tabindex="0"`. В этом случае элемент будет доступен при навигации через клавиатуру. Также `tabindex` позволяет менять порядок навигации, для этого необходимо установить положительное значение у `tabindex`, табуляция идет от меньшего к большему. Отрицательный `tabindex` исключает элемент из навигации.

7. Не стоит обнулять свойство `outline` у элементов интерфейса, это свойство позволяет показать какой элемент находится в данный момент в фокусе на странице.

8. Правильно задавайте атрибут `type` у тега `input`. Добавляете метки (теги `label`) для полей формы. Связывайте метки с полями формы через атрибут `for`.

9. Текстовые описания элементов интерфейса (меток, кнопок, ссылок и т.д.) должны быть понятными и уникальными. Например, такая ссылка `Правила соглашения пользователя можно посмотреть <a href="rules.html"> здесь</a>` является плохим примером, лучше сделать ссылку так `<a href="rules.html">Просмотреть правила соглашения пользователя</a>`. То есть лучше не использовать названия «Нажмите здесь», «Кнопка» и т.д.

10. Для таблиц необходимо задавать заголовки таблиц с помощью тега `caption`, а заголовки столбцов и, при необходимости, строк с помощью тега `th`. Это облегчит чтение таблиц скринридерами.

11. Для всех картинок необходимо задавать альтернативный текст (атрибут `alt`), также для корректной работы скринридера. Если картинка не имеет смысла (чисто декоративная), то оставляем атрибут `alt` пустым, но обязательно задаем, в этом случае скринридер ее пропустит при чтении, а иначе будет читать весть тег.

Стоит отметить, что соблюдение этих правил, позволят не только обеспечить доступность сайта, но и способствует SEO-оптимизации сайта. **SEO (Search Engine Optimization)** – это комплекс мер, которые должны

способствовать поднятию позиции сайта в выдача поисковых систем по определенным запросам. Конечная цель SEO – это увеличения сетевого трафика и последующая его монетизация, увеличение потенциальных клиентов.

Одним из современных механизмов обеспечения доступности является использование атрибутов ARIA (Accessible Rich Internet Applications – Доступные полнофункциональные интернет-приложения). ARIA – это набор атрибутов, которые могут быть добавлены к любым HTML-тегам, эти атрибуты поддерживаются большинством современных браузеров. Рассмотрим некоторые ARIA-атрибуты:

1. ARIA-роли (атрибут role) позволяет определить тип элемента (то есть чем он является) и указать для каких целей он служит (что он делает). Существует список ролей, с ним можно познакомится на сайте [2]. Выделяют четыре группы ролей:
   a. роли виджеты (`role="button"`, `role="menuitem"`, `role="slider"` и др.);
   b. композиционные роли (`role="listbox"`, `role="menu"`, `role="grid"` и др.);
   c. роли структуры документа (`role="article"`, `role="presentation"`, `role="group"` и др.);
   d. абстрактные роли (`role="alert"`, `role="window"`, `role="landmark"` и др.).

2. Aria-свойства определяют свойства элементов, которые можно использовать для придания им дополнительного значения или семантики. Примеры:
   a. свойство `aria-required="true"` укажет скринридеру, что элемент обязателен для заполнения
   b. свойство `aria-labelledby = "label"` позволяет поставить идентификатор на элемент, а затем ссылаться на него как на метку, в том числе на несколько элементов, например, два определения связываются с одним термином:

```
<dl>
    <dt id="tag">Тег</dt>
    <dd role="definition" aria-labelledby="tag">именованная метка</dd>
    <dd role="definition" aria-labelledby="tag">ключевое слово, по которому можно легко найти нужный материал</dd>
    …
</dl>
```

c. свойство `aria-label` создаёт текстовую метку текущего элемента в случае отсутствия видимого текста описания элемента, например, кнопка задана не текстом а символом, в этом случае лучше снабдить ее свойством `aria-label`:

```
<button aria-label="Закрыть" onclick="close()">X</button>
```

3. Aria-состояния определяют текущее состояние элемента. Примеры:
   a. атрибут `aria-disabled="true"` сообщает, что элемент формы не активен, его не нужно заполнять;
   b. атрибут `aria-hidden="true"` делает элемент невидимым для скринридера;
   c. атрибут `aria-checked="true"` указывает на текущее выбранное состояние checkbox-ов, radio-кнопок и других элементов;
   d. атрибут `aria-selected="true"` указывает на текущее выбранное состояние одного элемента (или нескольких) из списка.

Основные правило использования ARIA-атрибутов: не заменять ими существующие семантические теги (`nav`, `article` и т.д.) и атрибуты (`alt`, `for` и т.д.).

Кроссбраузерность – это способность веб-сайта одинаково (в идеале, идентично) отображаться и функционировать во всех популярных браузерах.

Список популярных на данный момент браузеров (как на ПК, так и на мобильных устройствах):

1. Chrome
2. Safari
3. Firefox
4. Microsoft Edge
5. Samsung Internet
6. Opera

В зависимости от целевой аудитории сайта и требований технического задания, может возникнуть необходимость ориентироваться и на другие браузеры (например, Internet Explorer), а также на разные версии одного и того же браузера.

Первое правило для обсечения кроссбраузерности – использовать только те технологии верстки (теги, атрибуты, CSS-свойства и т.п.), которые поддерживаются всеми браузерами, указанными в техническом задании. Проверить поддержку той или иной технологии можно с помощью сервиса Can I Use (https://caniuse.com). Пример проверки поддержки типа тега `<input type="tel">` в разных браузерах представлено на рисунке 130.

Рисунок 130 – Интерфейс сайта Can I Use

Зелёный цвет означает, что технология поддерживаются, красный – не поддерживается, серый – неизвестно. Так же могут быть некоторые уточнения или ограничения относительно поддержки тех или иных технологий. Например, в некоторых браузерах, чтобы то или иное CSS-свойство работало необходимо задавать ему ***префикс***.

Для включения в браузер экспериментальных свойств CSS, которые стандартом ещё не утверждены, разработчики браузеров могут добавить префикс к этому свойству и использовать его. Когда экспериментальное свойство утверждено в стандарте и прошло тестирование в браузере, у него обычно убирается префикс.

Примеры префиксов:

- -webkit-: браузеры Chrome, Safari, Opera;
- -moz-: браузер Mozilla Firefox;
- -ms-: браузер Internet Explorer;
- др.

Например, свойство box-sizing изначально не было стандартным и в браузерах Firefox (до 29 версии), Crome (до 10 версии), Safari (до версии 5.1) для него необходимо указывать префиксы соответствующих браузеров (рис. 131).

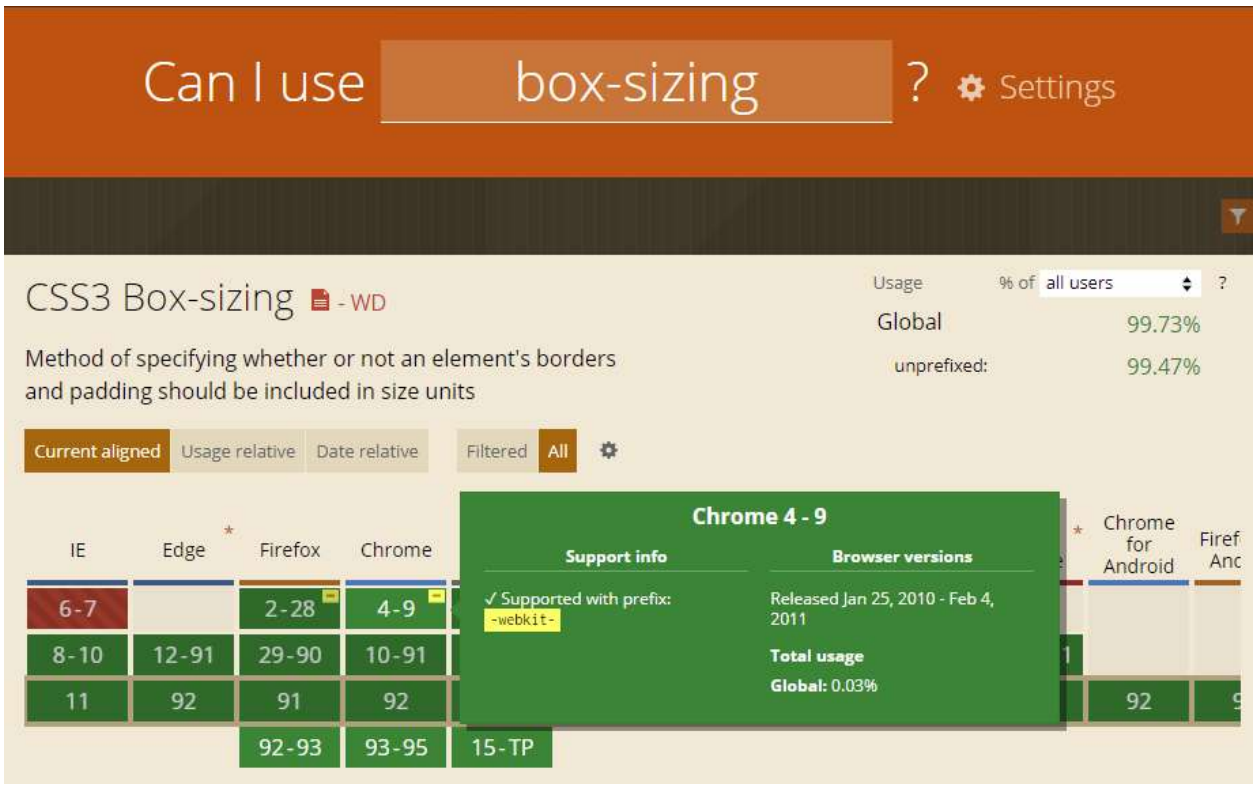


Рисунок 131 – Указание на необходимость добавления префиксов для CSS-свойства

Проверять «в ручную» каждое свойство и добавлять префиксы очень рутинная задача, поэтому есть системы Автопрефиксеры, которые позволяют по введенному CSS-коду получить все префиксы для всех необходимых CSS-свойств.

Другой способ решения проблем кроссбраузерности связан с использованием ***полифилов***. Полифил – это код, который позволяет реализовать какую-либо функциональность, не поддерживаемую в некоторых версиях веб-браузеров. Чаще всего полифилы написаны на языке программирования JavaScript и могут представлять собой целую библиотеку решений. Например, есть полифилы, которые позволяют корректно отображать в старых браузерах (например, Internet Explorer 9 версии и ниже) современные типы элементов `input` (`range`, `color`, `date` и т.д.). Или полифилы которые позволяют использовать медиа запросы в старых браузерах и т.д. Для большинства проблем в кроссбраузерной верстке можно найти полифил, который решает эту проблему.

2.10. Практическая работа 8. Инструменты проверки и тестирования сайта. Обеспечение доступности и кроссбраузерности

Цель работы: исправить верстку сайта для обеспечения доступности и кроссбраузерности.

Задачи работы:

1. Научиться работать с локальным сервером.
2. Ознакомиться с инструментами тестирования сайта.
3. Изучить роли, свойства и состояния ARIA.
4. Овладеть приемами обеспечения доступности сайтов.
5. Научиться использовать Автопрефиксер.

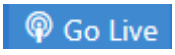
Инструменты: Visual Studio Code, браузер Google Chrome.

Задание 1. Установка локального сервера

Для просмотра сайта на разных устройствах, хотя бы в своей локальной сети, используется локальный сервер.

Локальный сервер – это эмулятор хостинга, программное обеспечение, которое позволяет сделать ваш компьютер веб-сервером. Это позволяет локально у себя на компьютере разрабатывать и тестировать полноценные клиент-серверные приложения.

В качестве локального сервера в данной практической работе будет использоваться расширение ***Live Server*** для VS Code.

1. Запустите VS Code.
2. Зайдите в раздел Расширения  .
3. Найдите расширение Live Server и установите его.
4. Откройте сайт из практической работы №7.
5. Запустите Live Server с помощью кнопки 

на панели состояния (справа внизу) или с помощью контекстного меню на странице index.html (рис. 132).

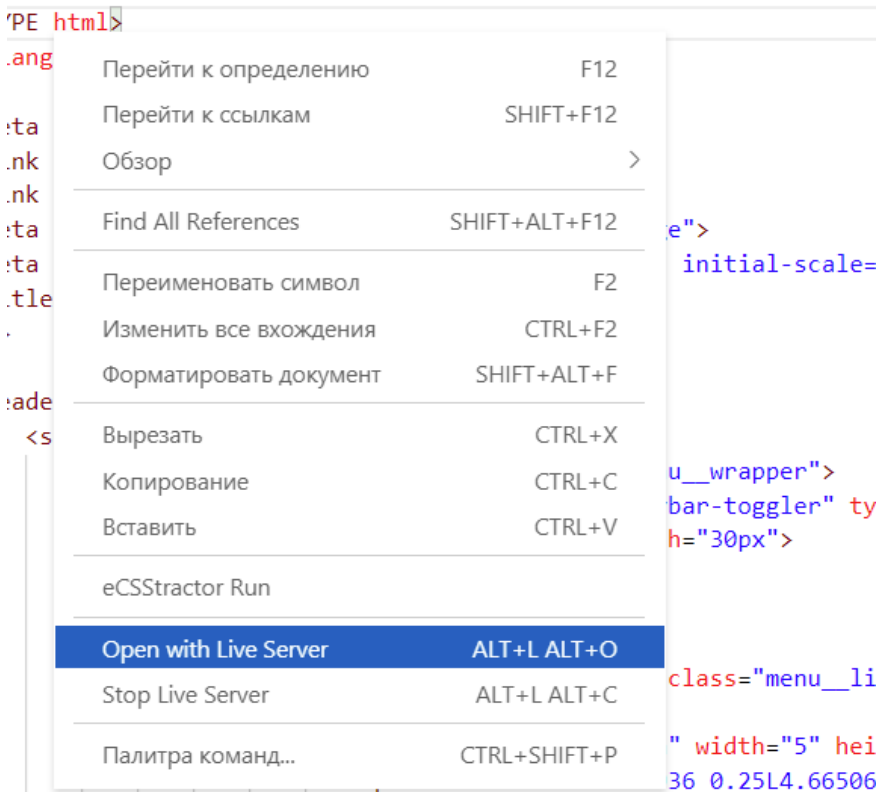


Рисунок 132 – Запуск Live Serverрез контекстное меню

6. В браузере откроется веб-страница. Просмотрите, но не закрывайте ее.

7. Внесите изменения в стили или в html-код. Сохраните изменения.

8. Перейдите в браузер с веб-страницей. Страница должна обновится автоматически без перезагрузки.

Теперь перезагрузка страниц будет проходить в реальном времени.

Задание 2. Просмотр веб-страниц на мобильных устройствах.

Если компьютер (с вашим сайтом и редактором VS Code) и мобильное устройство подключены к одной мобильной сети, то локальный сервер Live Server позволит просмотреть веб-страницу и на мобильном устройстве.

1. Откройте терминал в VS Code (меню Терминал -> Создать терминал). Терминал – это командная строка, встроенная в редактор кода.

2. Введите в терминал команду ipconfig и нажмите ENTER (рис. 133).

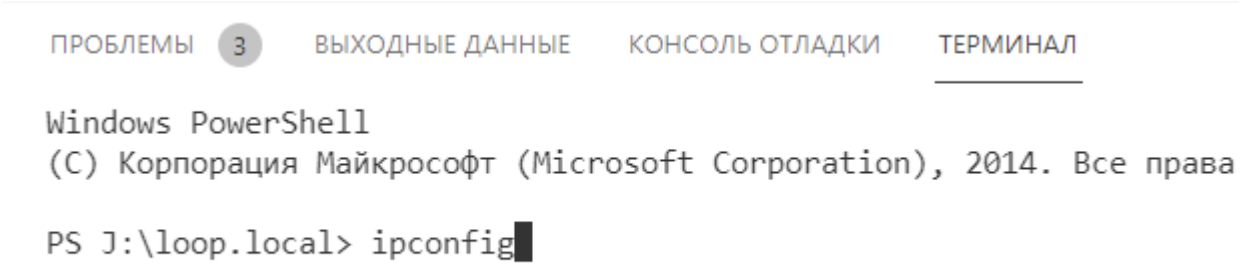


Рисунок 133 – Терминал VS Code

3. В терминале появится информация о настройках IP. Найдите информацию об IPv4-адресе. Запомните этот адрес (четыре числа).

4. Зайдите в браузер своего мобильного телефона и введите адрес страницы в следующем формате:

```
IPv4-адрес:5500/index.html
```

Например:

```
10.10.10.32:5500/index.html
```

5500 – это порт, вместо index.html могут быть и другие веб-страницы, которые находятся в проекте.

Теперь вы можете просматривать сайт на любых устройствах, которые подключены к вашей wifi-сети.

5. Протестируйте сайт на мобильных устройствах – выполните несколько пользовательских сценариев (навигация, работа с формой и т.д.).

6. Оцените удобство работы с вашим сайтом на мобильном устройстве. При необходимости исправьте верстку сайта.

Задание 3. Тестирование сайта с помощью сервисов

1. Проверьте валидацию кода с помощью ***валидатора*** на сайте https://www.w3.org/. Зайдите на сайт и в меню слева найдите пункт Validators. Зайдите на эту страницу.

2. Зайдите на вкладку Validate by Direct Input (рис. 134).

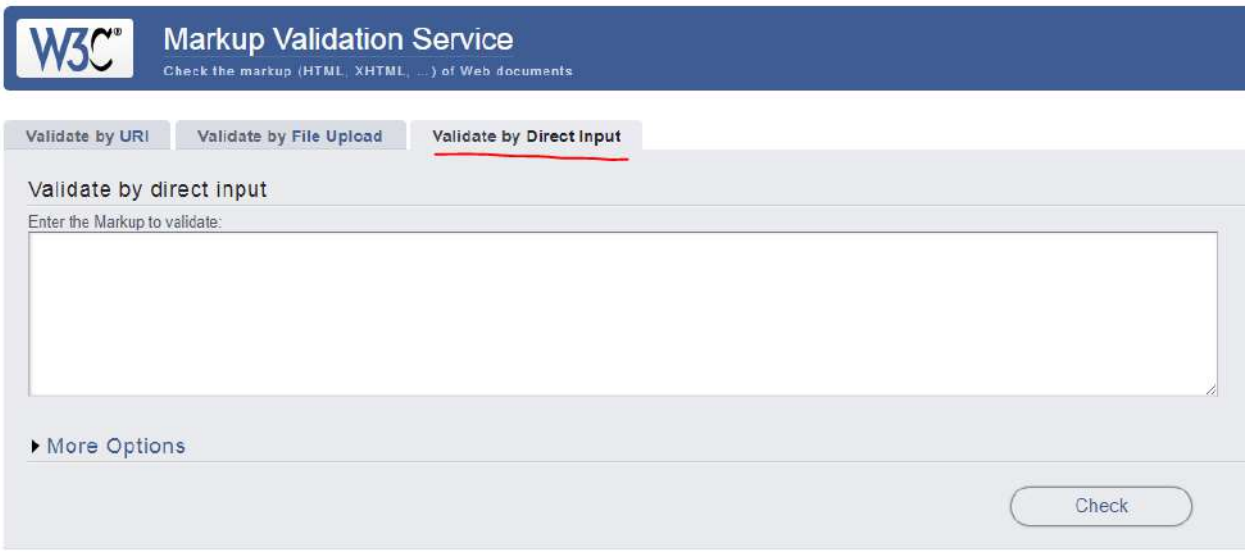


Рисунок 134 – Валидатор кода

3. Скопируйте и вставьте код страницы index.html в раздел Enter the Markup to validate.

4. Запустите валидацию – нажмите кнопку Check.

5. Просмотрите список ошибок и предупреждений. Исправьте их в коде сайта.

6. Проведите валидацию повторно.

7. Проведите валидацию CSS кода. Для этого выберите CSS валидатор (рис. 135).

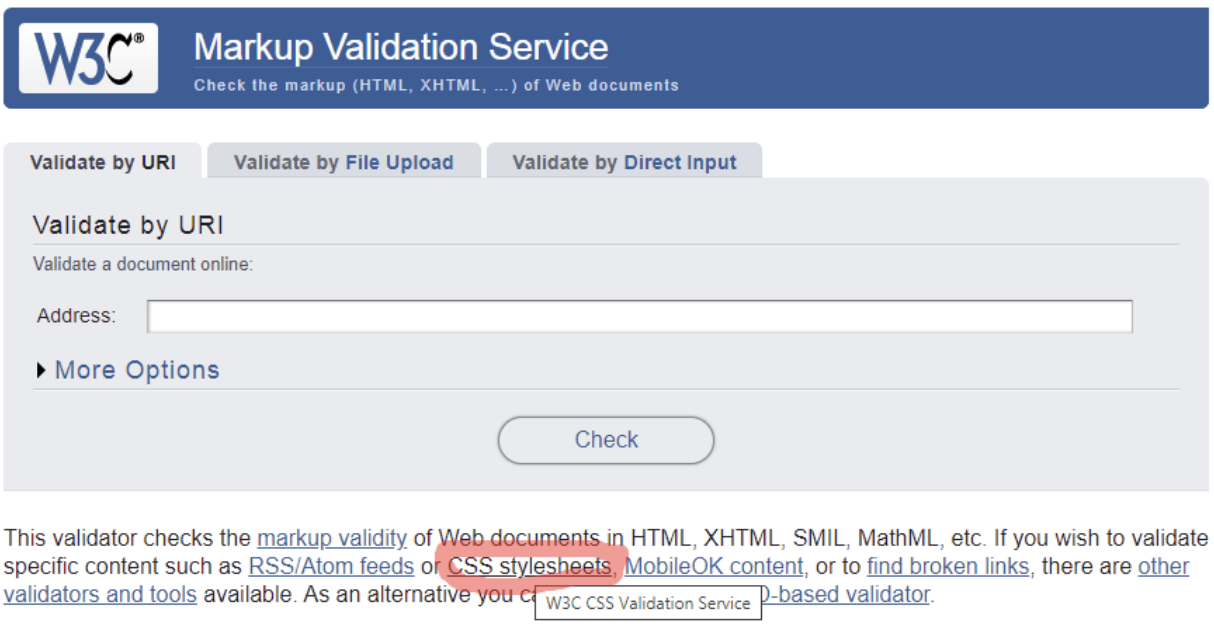


Рисунок 135 – Выбор CSS валидатора

8. Аналогично скопируйте и загрузите CSS-код вашего сайта.
9. Запустите проверку.
10. Просмотрите и исправьте ошибки и предупреждения.

Данный способ валидации не очень удобен. Установим расширение для проверки валидации сразу в редакторе VS Code.

11. Установите расширение ***W3C Web Validator***.
12. Перейдите на html или css страницу и запустите валидатор с помощью кнопки 

на панели состояния (слева внизу).

Теперь вы сможете проверять код сразу в редакторе.

13. Запустите Live Server для вашей веб-страницы.
14. Откройте инструменты разработчика в браузере Chrome.
15. Перейдите на вкладку LightHouse. ***LightHouse*** – это инструмент автоматического тестирования веб-приложений.
16. Установите настройки как на рисунке 136 и нажмите кнопку Generate Report.

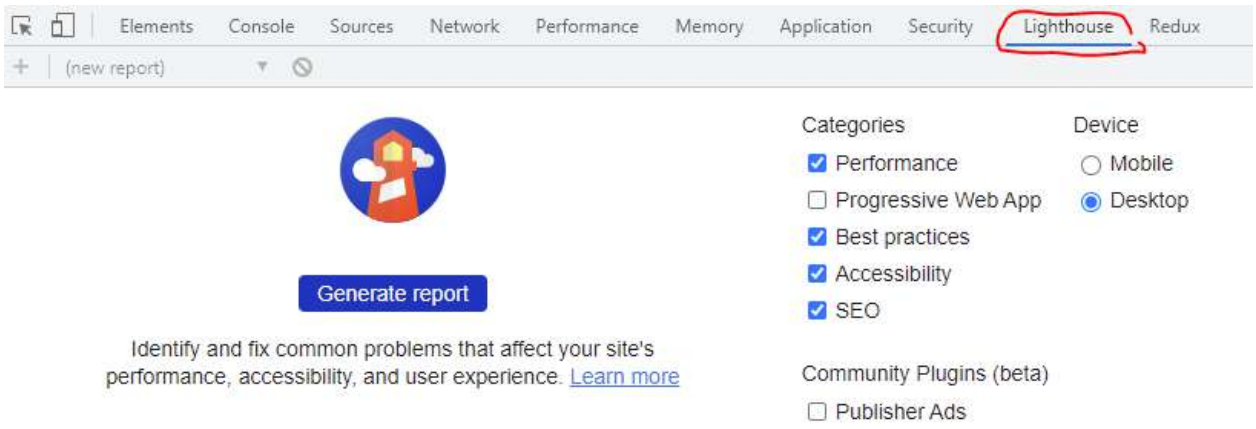


Рисунок 136 – Вкладка Light House

17. Подождите загрузки отчета (рис. 137). Ваш отчет может отличаться от представленного на рисунке.

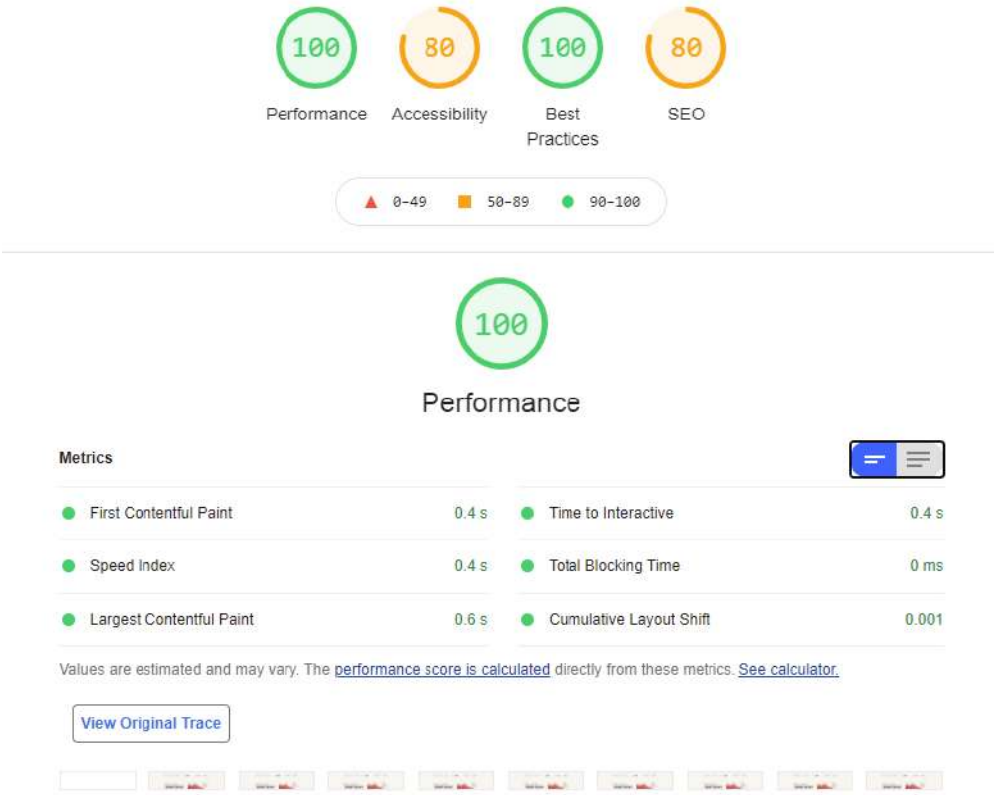


Рисунок 137 – Пример отчета LightHouse

18. Изучите все разделы отчета: Производительность (Performance), Доступность (Accessibility), Лучшие практики (Best Practices) и SEO.

19. Просмотрите ошибки (красный треугольник) и замечания (желтый квадрат). По возможности исправьте их и перезагрузите отчет.

20. Сформируйте отчет в LightHouse для мобильных устройств.

21. Проведите еще один тест доступности – реализуйте все пользовательские сценарии на сайте используя только клавиатуру.

Одним из самых популярных ***инструментов кроссбраузерного тестирования*** является сервис http://www.browserstack. Данный сервис использует реальные устройства для тестирования и поддерживает огромное число разных версий браузеров на разных операционных системах. Кроме тестирования сайтов, опубликованных в сети Интернет, сервис позволяет тестировать локальные сайты (сайты, расположенные на локальном сервере вашего компьютера). Но данный инструмент является платным с ежемесячной подпиской.

Задание 4. Работа с ARIA

1. Установите расширение Screen Reader для Chrome.

2. Зайдите на страницу сайта и включите расширение. Прослушайте сайт.

3. Изменим верстку изображения корзины в шапке сайта –  . Сделаем ее кнопкой, картинку назначим кнопке через стили.

HTML-код:

```
<li class="basket__item">
    <button class="btn-basket"></button>
</li>
```

CSS-код:

```
.btn-basket{
    width: 50px;
    height: 50px;
    background: url('../img/shopping-basket-button.svg') no-repeat;
    border: none;
    cursor: pointer;
}
```

4. Если запустить скринридер, то кнопку корзины он озвучит как «кнопка».

5. Добавьте кнопке свойство aria-label="Корзина". Теперь скринридер озвучит кнопку как «Корзина».

6. Проверять доступность элементов можно с помощью вкладки Accissibility в инструментах разработчика.

Часто при стилизации элементов формы (радио кнопок, чекбоксов и т.д.) нарушается их семантическая (смысловая) верстка. Это приводит к некорректному чтению этих элементов скринридерами. Большинство проблем уже имеют решения.

7. Зайдите на официальный сайт, в раздел практик использования ARIA https://www.w3.org/TR/wai-aria-practices.

8. Найдите раздел использования Radio Group и просмотрите пример.

9. Также просмотрите пример реализации Меню с выпадающими списками.

Задание 5. Работа с префиксами

1. Зайдите на сайт https://autoprefixer.github.io/ru/.

2. Скопируйте CSS-код проекта и вставьте в левое окно автопревиксера (рис. 138).

Рисунок 138 – Автопрефиксер

3. Скопируйте результат работы автопрефиксера (правое окно) и замените им CSS-код в проекте.

Задания для самостоятельного выполнения

1. Проверьте сайт по следующим пунктам:
   a. у всех тегов `img` есть атрибут `alt`;
   b. у элементов ввода и выбора информации на формах есть либо метки (тег `label` с атрибутом `for`) или `aria-labelledby`.
2. Проведите все виды тестирования сайта, которые вы знаете.

Заключение

Существуют разные подходы к созданию веб-сайтов и приложений: написание кода «с нуля», создание сайтов с помощью системы правления контентом сайта или создание приложений с помощью фреймворков. Но любой подход требует знаний основ веб-технологий, ключевых технологий верстки веб-страниц – язык разметки гипертекста HTML и каскадные таблицы стилей CSS. Эти технологии являются базовыми для всего направления веб-технологий.

HTML позволяет создать структуру страницы, построить ее каркас и наполнить его контентом. CSS отвечает за внешний вид заданных элементов страницы. Вместе эти две технологии позволяют разработать модульную сетку сайта, отвечающую требованиям кроссбраузерности, адаптивности и доступности, оформить контент сайта, добавить элементам интерактивность, полностью сформировать интерфейс сайта.

При изучении тем данного пособия вы научились:

– верстать веб-страницы с учетом Search Engine Optimization;

– проверять корректность и исправлять HTML и CSS код с помощью инструментов разработчика.

– создавать модульные сетки с помощью flexbox;

– корректно использовать CSS для обеспечения кроссбраузерности;

– писать медиа запросы и создавать адаптивные сайты;

– обеспечивать доступность и кроссбраузерность сайтов.

В дальнейшем для углубления и расширения навыков в области верстки сайтов стоит рассмотреть следующие вопросы:

– работа с SVG-форматом и спрайтами;

– создание двумерных трансформаций, переходов, CSS-фильтров, CSS-функций;

– использование препроцессоров CSS, постпроцессоров CSS и шаблонизаторов HTML;

– оптимизация сайтов;

– автоматизация процесса разработки и сборка проектов;

– использование библиотек и фреймворков для создания веб-интерфейсов.

Вопросы для самопроверки

1. Сеть Интернет. Возможности сети Интернет.
2. Информационные услуги сети Интернет.
3. Организация веб-сайтов.
4. Современные технологии разработки веб-сайтов.
5. Язык HTML как средство создания веб-страниц.
6. Основные элементы языка HTML. Теги и контейнеры.
7. Структура документа на языке HTML.
8. Оформление текста в HTML. Создание списков.
9. Работа с изображениями HTML.
10. Создание ссылок в HTML. Виды ссылок.
11. Теги для создания таблиц.
12. Формы в HTML. Элементы форм. Типы тега input.
13. Раздел head. Meta-данные страницы.
14. Кодстайл HTML.
15. Каскадные таблицы стилей CSS.
16. Стилевые правила. Способы задания стилей.
17. Селекторы. Идентификаторы и классы.
18. Селекторы: дочерние, потомков, соседние.
19. Псевдо элементы и псевдо классы.
20. Селекторы атрибутов.
21. Стили для форматирования текста.
22. Стили позиционирования и размера.
23. Стили для определения фона и цвета.
24. Каскадирование стилей. Приоритеты CCS-правил. Специфичность.
25. Строчные и блочные элементы. Свойство display, его типы.
26. Позиционирование элементов.
27. Стилизация и валидация форм.
28. Параметры шрифтов. Единицы измерения размеров.
29. Подгружаемые шрифты.
30. Кодстайл CSS.
31. Построение шаблона. Сетки.
32. Построение сеток с помощью Flexbox.
33. Flex-контейнер. Свойства flex-контейнера.
34. Flex-элементы. Свойства flex-элементов.
35. Медиа запросы. Правила адаптивной верстки.
36. Доступность веб-сайтов. Инструменты доступности.

37.ARIA роли, свойства и состояния.
38.Кроссбраузерность сайтов. Префиксы и полифилы.
39.Редакторы кода для создания сайта.
40.Редактор Visual Studio Code. Надстройка Emmet.
41.Инструменты разработчика в браузере.
42.Инструменты тестирования сайтов.

Темы для углубленного изучения

1. Формат масштабируемой векторной графики SVG. Спрайты.
2. Построение сеток с помощью GRID.
3. CSS-трансформации.
4. Создание анимации с помощью CSS. Свойства transition. Keyframe.
5. Добавление аудио и видео на страницу.
6. CSS-функции и CSS-фильтры. Постпроцессоры.
7. Препроцессоры CSS.
8. Шаблонизаторы HTML.
9. UI фреймворки и библиотеки.
10. Верстка HTML-писем.
11. Отрисовка сайтов в браузере – конвейер визуализации пикселей.
12. Оптимизация сайтов. Минимизация файлов CSS, HTML. Оптимизация шрифтов и изображений.
13. Автоматизация и сборка проектов по верстке сайтов.
14. Менеджер задач Gulp.
15. Хостинг.
16. Системы управления контентом сайтов.

Задание для выполнения проектов

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.

Полезные сервисы и Интернет-ресурсы

1. Can I Use [Электронный ресурс]: таблицы поддержки браузеров для современных веб-технологий. — Режим доступа: https://caniuse.com.— Загл. с экрана.
2. CSS Minifier [Электронный ресурс]: минификатор CSS. — Режим доступа: https://cssminifier.com. — Загл. с экрана.
3. CSS-Tricks [Электронный ресурс]: ресурс о создании сайтов. — Режим доступа: https://css-tricks.com.— Загл. с экрана.
4. Emmet [Электронный ресурс]: официальный сайт. — Режим доступа: https://www.figma.com.— Загл. с экрана.
5. Figma [Электронный ресурс]: графический онлайн-редактор. — Режим доступа: https://www.figma.com.— Загл. с экрана.
6. Flexbox Froggy [Электронный ресурс]: обучающая игра по Flexbox. — Режим доступа: https://flexboxfroggy.com.— Загл. с экрана.
7. Font awesome [Электронный ресурс]: библиотека иконок. — Режим доступа: https://fontawesome.com.— Загл. с экрана.
8. Font Squirrel [Электронный ресурс]: сервис подбора и генерации шрифтов. — Режим доступа: https://www.fontsquirrel.com. — Загл. с экрана.
9. Google Fonts [Электронный ресурс] — Режим доступа: https://fonts.google.com/. — Загл. с экрана.
10. HTML Academy [Электронный ресурс]: интерактивные онлайн-курсы. — Режим доступа: https://htmlacademy.ru. — Загл. с экрана.
11. MDN Web Docs [Электронный ресурс]: resources for developers, by developers. — Режим доступа: https://developer.mozilla.org. — Загл. с экрана.
12. Squoosh [Электронный ресурс]: сервис оптимизации изображений. — Режим доступа: https://squoosh.app. — Загл. с экрана.
13. Visual Studio Code [Электронный ресурс]: официальный сайт. — Режим доступа: https://code.visualstudio.com. — Загл. с экрана.
14. WebReference [Электронный ресурс]: ресурс о веб-технологиях. — Режим доступа: https://webref.ru.— Загл. с экрана.
15. Автопрефиксер CSS онлайн [Электронный ресурс]. — Режим доступа: https://autoprefixer.github.io/ru. — Загл. с экрана.
16. Веб-стандарты [Электронный ресурс]: сообщество разработчиков. — Режим доступа: https://web-standards.ru.— Загл. с экрана.
17. Горячие клавиши Visual Studio Code [Электронный ресурс]. — Режим доступа: https://nikomedvedev.ru/other/vscodeshortcuts/hotkeys.html. — Загл. с экрана.

18. Консорциум всемирной паутины [Электронный ресурс]: официальный сайт. — Режим доступа: https://www.w3.org. — Загл. с экрана.

19. Colorscheme.ru [Электронный ресурс]: Инструмент для подбора цветов и генерации цветовых схем. — Режим доступа: https://colorscheme.ru/html-colors.html. — Загл. с экрана.

Библиографический список

1. Htmlbook [Электронный ресурс] — Режим доступа: http://htmlbook.ru/. — Загл. с экрана.

2. MDN Web Docs [Электронный ресурс]: resources for developers, by developers. — Режим доступа: https://developer.mozilla.org. — Загл. с экрана.

3. Oxford Languages [Электронный ресурс]: словарь русского языка. — Режим доступа: https://languages.oup.com/google-dictionary-ru/. — Загл. с экрана.

4. WebReference [Электронный ресурс]: ресурс о веб-технологиях. — Режим доступа: https://webref.ru.— Загл. с экрана.

5. Консорциум всемирной паутины [Электронный ресурс]: официальный сайт. — Режим доступа: https://www.w3.org. — Загл. с экрана.

6. Расчёт итогового размера с flex-grow [Электронный ресурс]: GitHub Gist. — Режим доступа: https://gist.github.com/monochromer/ab1034528c72c35e96fa01a236739589. — Загл. с экрана.