

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

ИНСТИТУТ НЕПРЕРЫВНОГО И ДИСТАНЦИОННОГО ОБРАЗОВАНИЯ

КАФЕДРА КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ И ПРОГРАММНОЙ ИНЖЕНЕРИИ

ОЦЕНКА

ПРЕПОДАВАТЕЛЬ

старший преподаватель		Н.А. Соловьева
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ

Применение каскадных таблиц стилей

по дисциплине: Web-технологии

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ гр. №	Z1431		М.Д.Быстров
	номер группы	подпись, дата	инициалы, фамилия

Студенческий билет №	2021/3572
----------------------	-----------

Санкт-Петербург 2024

Оглавление

Оглавление	2
Задание	3
Средства, использованные при выполнении работы	4
Результат выполнения базового задания	5
Результат выполнения дополнительного задания	6
Скриншоты.....	8
Исходный код программы	14
Вывод.....	38

Задание

Таблица 1 Вариант индивидуального задания

№	Гр z1431	Тема сайта	Вариант таблицы и списка
1	Быстров		1

Таблица 2 Тема в соответствии с вариантом

№ варианта	Тема
1	Язык Котлин

Таблица 3 Вид оформления таблицы и списка в соответствии с вариантом

№	Оформление таблицы	Оформление списков
1	Прокрутка в таблице	Разные типы цифр

Средства, использованные при выполнении работы

1. Редактор Visual Studio Code
2. Браузер Mozilla Firefox

Результат выполнения базового задания

HTML-страницы, разработанные в рамках лабораторной работы № 1, оформлены с применением каскадных таблиц стилей.

1) Использованы три варианта подключения таблиц css:

- связные таблицы стилей (отдельный внешний файл) - листинг на стр.

22

- глобальные таблицы стилей (блок css в файле html (тег style)) рис.

листинг на стр. 14

- локальные таблицы стилей (локально для одного тега (атрибут style))

листинг на стр. 18.

2) В таблицах

- оформлены границы - рис. №4, листинг на стр. 23;

- в одну из ячеек вставлена картинка, при этом сохранено

выравнивание в таблице - рис. №4, листинг на стр. 24;

3) Использованы следующие технические средства:

- селекторы: тегов, классов, идентификаторов, составной, листинг на стр. 19-22, 22-24;

- псевдоклассы (:hover) - рис. №2, листинг на стр. 21 (навигация)

- указание размера: в пикселях, в миллиметрах, через процент, листинг на стр. 19,24,19;

- указание цвета: слово, шестнадцатеричный формат, десятичный формат - листинг на стр. 21,19,20.

4) Добавлена прокрутка в таблице - рис. №4, листинг на стр. 22-23;

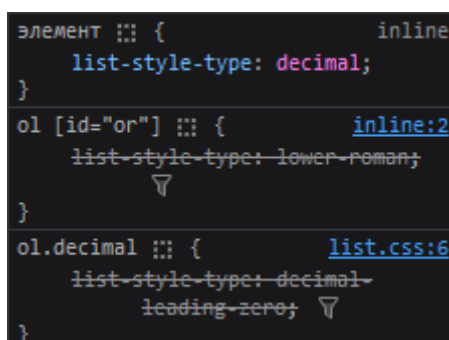
5) Использованы разные типы цифр в списке - рис. №3, листинг на стр. 22.

Результат выполнения дополнительного задания

1. Для одного и того же элемента применены правила, расположенные в блоках разного уровня (связанные, глобальные, локальные).

При использовании стилей наивысший приоритет имеют локальные стили, затем глобальные, и в последнюю очередь – связанные.

Это продемонстрировано на примере списка, к которому применено три разных стиля `list-style-type`. Как видно в консоли браузера, применяется тот стиль, который определен локально.



```
элемент ::: {                               inline
  list-style-type: decimal;
}
ol [id="or"] ::: {                           inline:2
  list-style-type: lower-roman;
}
ol.decimal ::: {                             list.css:6
  list-style-type: decimal;
  leading-zero;
}
```

Рисунок 1 Применение стилей из разных источников

Листинг – страницы 14,18,22. Рис. №4.

2. использован символ «+» для объединения селекторов - рис. №2, листинг на стр. 22;

3. использован псевдоэлемент (`:first-letter`, `:first-line` и т.д.) - рис. №7, листинг на стр. 37;

4. в оформлении применены `margin`, `border`, `padding`. Разница состоит в том, что `margin` – внешний отступ, `padding` – внутренний отступ, `border` – толщина и стиль границы элемента. Рис. №4 (таблица – `border`, абзацы – `margin`, контент – `padding`), листинг на стр. 30,31,23;

5. скруглены углы прямоугольного элемента (свойство `border-radius`) - рис. №7, листинг на стр. 23;

6. сделан фон с градиентом (свойство `background-image: xxx-gradient`) - рис. №3 (футер), листинг на стр. 21;

7. использовано свойство text-decoration - рис. №2, листинг на стр. 21;
8. применен селектор атрибута - рис. №3, листинг на стр. 12.

Скриншоты

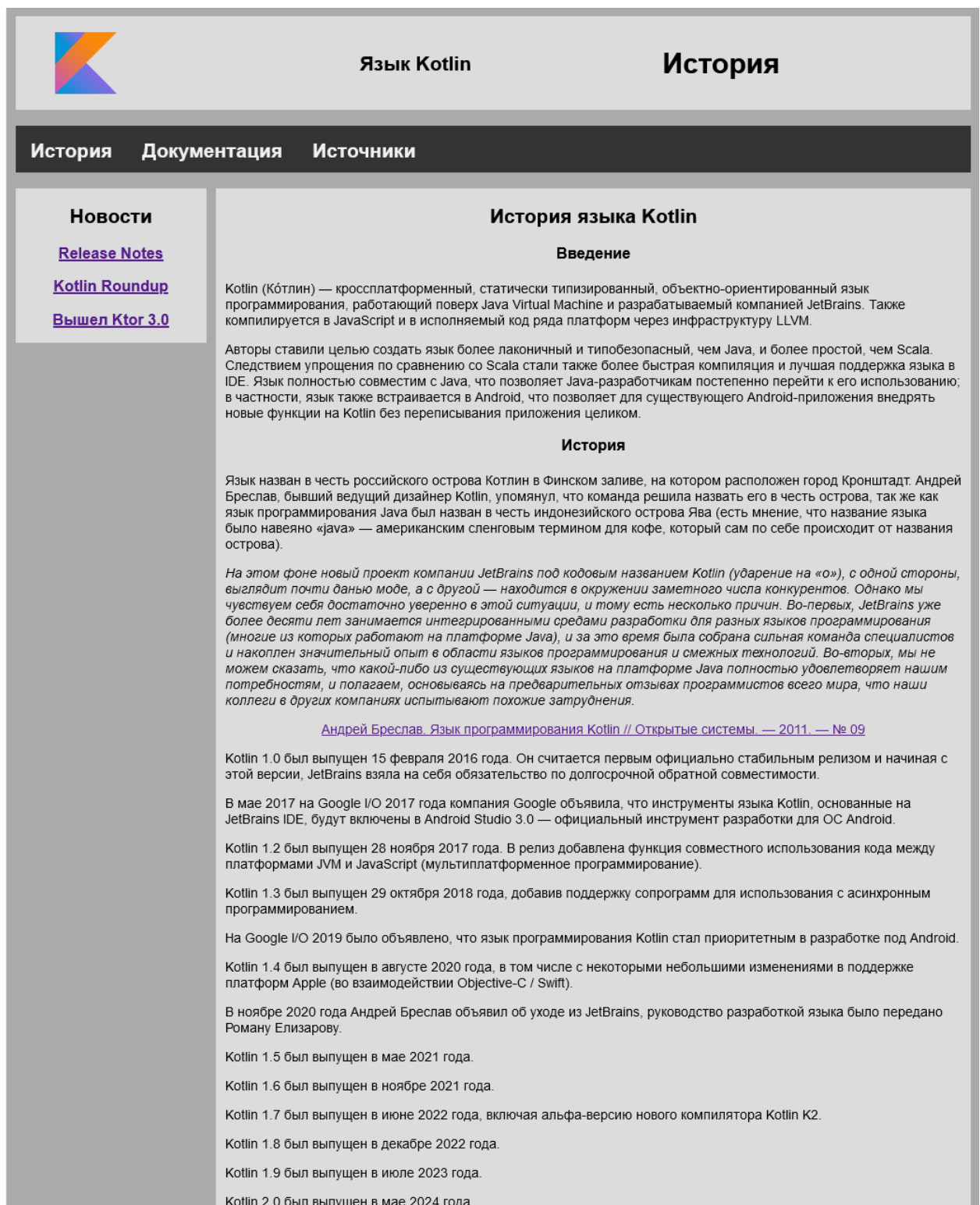


Рисунок 2 Фрагмент первой страницы сайта

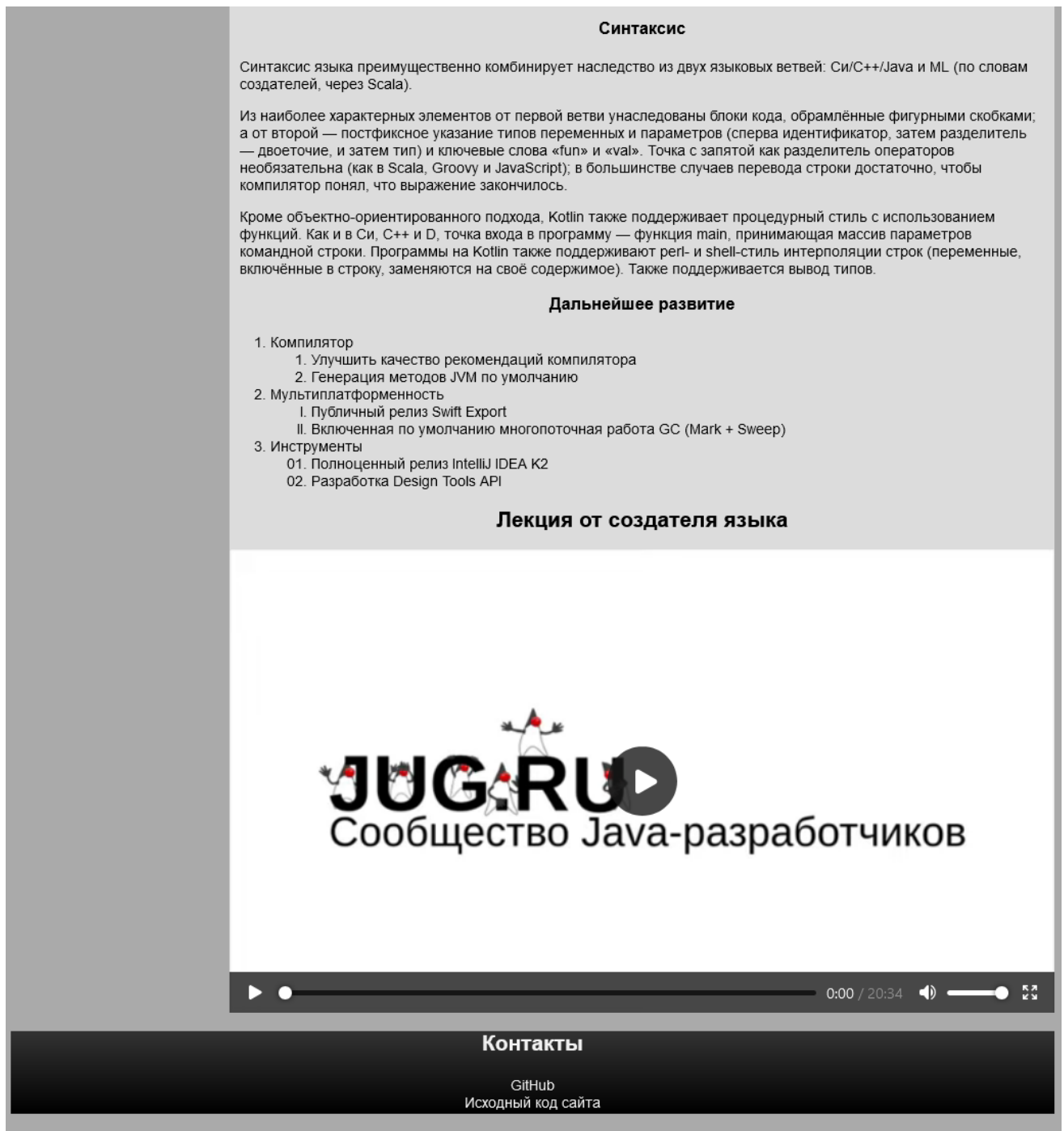


Рисунок 3 Фрагмент первой страницы сайта



Новости

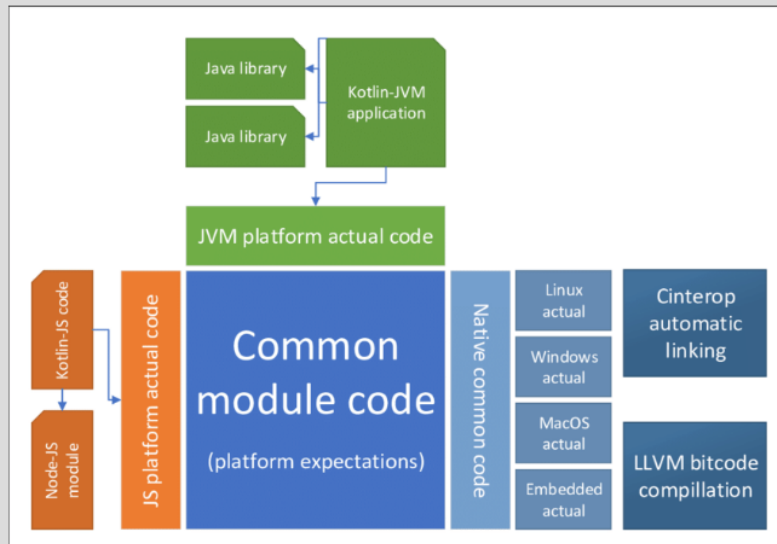
[Release Notes](#)[Kotlin Roundup](#)[Вышел Ktor 3.0](#)

Статус поддержки последних версий

Описание поддержки версий языка Kotlin				
Версия	1.7	1.8	1.9	2.0
Обновления безопасности	Нет			Да
Дата выпуска	09 Jun 2022	28 Dec 2022	06 Jul 2023	21 May 2024

Kotlin Multiplatform

Поддержка мультиплатформенного программирования является одним из ключевых преимуществ Kotlin. Она сокращает время, затрачиваемое на написание и поддержку одного и того же кода для разных платформ, сохраняя при этом гибкость и преимущества нативного программирования.



Примеры использования Kotlin Multiplatform

Android и iOS приложения



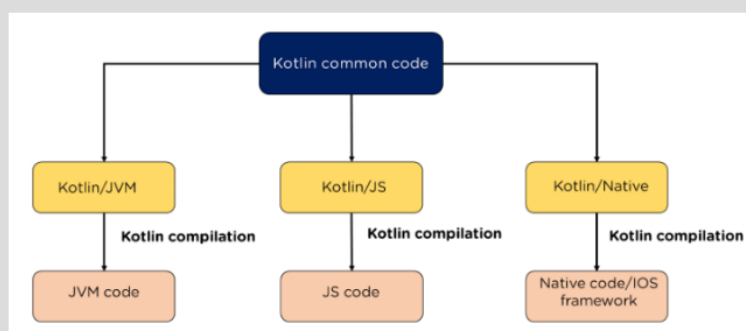
Использование **одного кода** на всех мобильных платформах является одним из основных вариантов использования Kotlin Multiplatform. С помощью Kotlin Multiplatform Mobile вы можете создавать мультиплатформенные мобильные приложения, совместно использующие один код и в Android, и в iOS, например бизнес-логику, возможности

Рисунок 4 Фрагмент второй страницы сайта



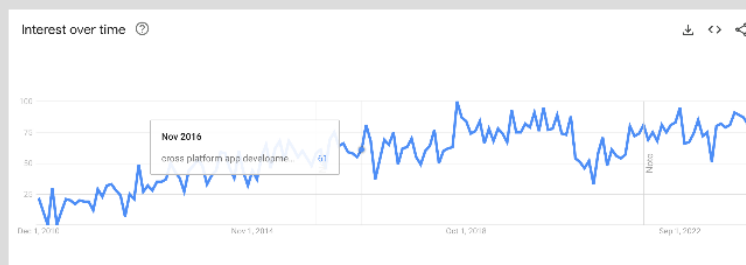
Другой сценарий, когда совместное использование кода может принести пользу, — это приложение, в котором логика может быть повторно использована как на стороне сервера, так и на стороне клиента, запущенного в браузере. С этим Kotlin Multiplatform тоже прекрасно справляется.

Мультиплатформенные библиотеки



Kotlin Multiplatform также отлично подойдет для авторов библиотек. Вы можете создать мультиплатформенную библиотеку с общим кодом и ее платформенными реализациями для *JVM*, *JS* и собственных платформ. После публикации мультиплатформенная библиотека может использоваться в других кроссплатформенных проектах в качестве зависимости.

Общий код для мобильных и веб-приложений



Еще одним популярным примером использования Kotlin Multiplatform является совместное использование одного и того же кода в Android, iOS и веб-приложениях. Это сокращает объем бизнес-логики, написанной фронтенд-разработчиками, и помогает более эффективно внедрять продукты, сокращая затраты на разработку и тестирование.

Определение имени пакета и импорт

Имя пакета указывается в начале исходного файла, так же как и в Java.

```
package my.demo

import java.util.*

// ...
```

Но в отличие от Java, нет необходимости, чтобы структура пакетов совпадала со структурой папок: исходные файлы могут располагаться в произвольном месте на диске.

Рисунок 5 Фрагмент страницы 2

Точка входа в программу

В Kotlin точкой входа в программу является функция main.

```
fun main() {  
    println("Hello world!")  
}
```

Другая форма main может принимать массив строк String.

```
fun main(args: Array) {  
    println(args.contentToString())  
}
```

Вывод в стандартный поток

print выводит свой аргумент в стандартный поток вывода.

```
print("Hello")  
print("world!")
```

println выводит свой аргумент и добавляет перевод строки, так что следующее, что вы выведете, появится на следующей строке.

```
println("Hello world!")  
println(42)
```

Функции

Функция принимает два аргумента Int и возвращает Int.

```
fun sum(a: Int, b: Int): Int {  
    return a + b  
}
```

В качестве тела функции может выступать выражение. Тогда тип возвращаемого значения определяется автоматически.

```
fun sum(a: Int, b: Int) = a + b
```

Функция, не возвращающая никакого значения (void в Java).

```
fun printSum(a: Int, b: Int): Unit {  
    println("сумма $a и $b равна ${a + b}")  
}
```

Тип возвращаемого значения Unit может быть опущен.

```
fun printSum(a: Int, b: Int) {  
    println("сумма $a и $b равна ${a + b}")  
}
```

Переменные

Неизменяемые (только для чтения) локальные переменные определяются с помощью ключевого слова val. Присвоить им значение можно только один раз.

```
val a: Int = 1 // Инициализация при объявлении  
val b = 1 // Тип `Int` определен автоматически  
val c: Int // Указывать тип обязательно,  
// если переменная не инициализирована сразу  
c = 1 // Последующее присвоение
```

Изменяемые переменные объявляются с помощью ключевого слова var.

```
var x = 5 // Тип `Int` определен автоматически  
x += 1
```

Вы можете объявлять глобальные переменные.

```
val PI = 3.14  
var x = 0  
  
fun incrementX() {  
    x += 1  
}
```

Контакты

GitHub

Исходный код сайта

Рисунок 6 Фрагмент страницы 2

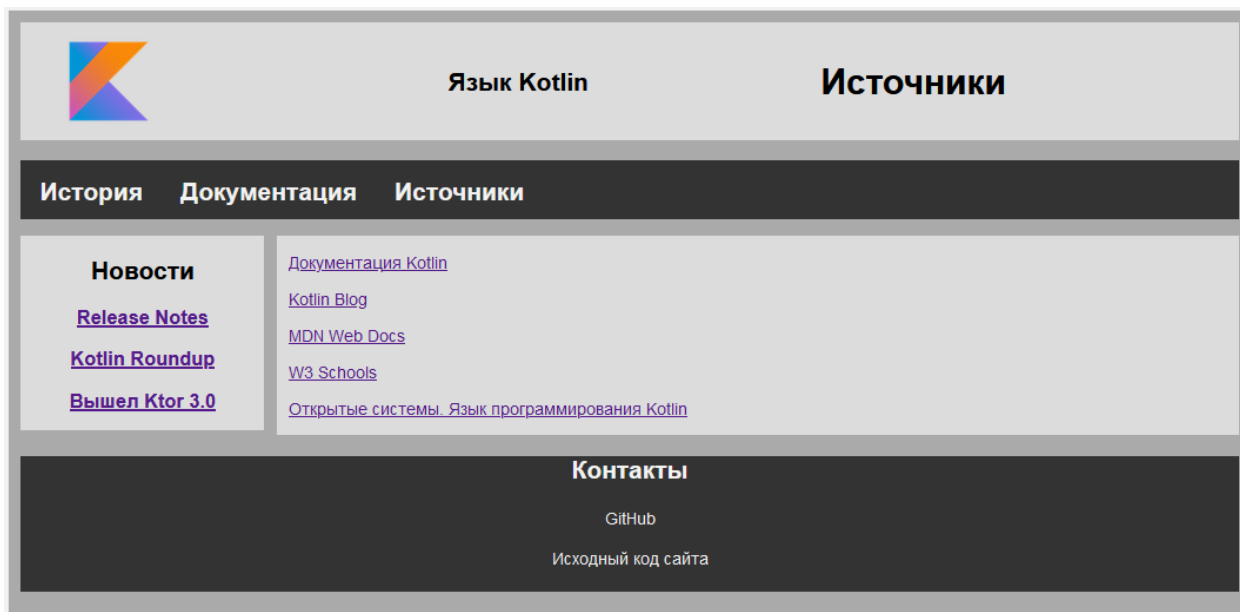


Рисунок 7 Страница 3

Исходный код программы

1. Файл "1.html"

```
<html>

<head>
  <title>Язык Kotlin</title>
  <link rel="shortcut icon" type="image/x-icon" href="img/kotlin_logo.png" />
  <link rel="stylesheet" type="text/css" href="1.css">
  <link rel="stylesheet" type="text/css" href="list.css">
  <meta charset="UTF-8">
  <!-- meta - дополнительное задание -->
  <meta name="description" content="Kotlin programming language">
  <meta name="keywords" content="Kotlin, Java, JVM, JDK">
  <meta name="author" content="Maxim Bystrov">

  <style>
    ol [id="or"]{
      list-style-type: lower-roman;
    }
  </style>
</head>

<body>
  <div id="container" class="container">

    <!-- шапка - обязательный элемент сайта -->
    <div id="header">

      <div id="logo"></div>

      <h2 id="sitename">Язык Kotlin</h2>

      <h3 id="pagename">История</h3>

    </div>

    <!-- навигация между страницами - обязательный элемент сайта -->
    <div id="navigation" class="container">
      <h2><a href="1.html">История</a>
        <a href="2.html">Документация</a>
        <a href="3.html">Источники</a>
      </h2>
    </div>
```

```

<br>

<!-- боковая панель -->
<div id="sidebar">
  <h2>Новости</h2>
  <h3><a href="https://kotlinlang.org/docs/releases.html">Release
Notes</a></h3>
  <h3><a href="https://blog.jetbrains.com/kotlin/2024/11/kotlin-
roundup-kodee-s-top-picks/">Kotlin Roundup</a>
  </h3>
  <h3><a href="https://blog.jetbrains.com/kotlin/2024/10/ktor-3-
0/">Вышел Ktor 3.0</a></h3>
</div>

<!-- основной контент -->
<div id="content">
  <h2>История языка Kotlin</h2>

  <h3>Введение</h3>
  <p>
    Kotlin (Кóтлин) – кроссплатформенный, статически типизированный,
    объектно-ориентированный язык программирования, работающий поверх
    Java Virtual Machine и разрабатываемый
    компанией JetBrains.
    Также компилируется в JavaScript и в исполняемый код ряда
    платформ через инфраструктуру LLVM.
  </p>
  <p>
    Авторы ставили целью создать язык более лаконичный и
    типобезопасный, чем Java, и более простой, чем
    Scala. Следствием упрощения по сравнению со Scala стали также
    более быстрая компиляция и лучшая
    поддержка языка в IDE. Язык полностью совместим с Java, что
    позволяет Java-разработчикам постепенно
    перейти к его использованию; в частности, язык также встраивается
    в Android, что позволяет для
    существующего Android-приложения внедрять новые функции на Kotlin
    без переписывания приложения целиком.
  </p>

  <h3>История</h3>
  <p>
    Язык назван в честь российского острова Котлин в Финском заливе,
    на котором расположен город Кронштадт. Андрей Бреслав, бывший
    ведущий дизайнер Kotlin, упомянул, что команда решила назвать его
    в честь острова, так же как язык программирования Java был назван
    в честь индонезийского острова Ява (есть мнение, что название
    языка было навеяно «java» – американским
    сленговым термином для кофе,

```

который сам по себе происходит от названия острова).

</p>

<p>

<!-- дополнительное задание - cite -->

<cite>

На этом фоне новый проект компании JetBrains под кодовым названием Kotlin (ударение на «о»), с одной стороны, выглядит почти данью моде, а с другой – находится в окружении заметного числа конкурентов. Однако мы чувствуем себя достаточно уверенно в этой ситуации, и тому есть несколько причин. Во-первых, JetBrains уже более десяти лет занимается интегрированными средами разработки для разных языков программирования (многие из которых работают на платформе Java), и за это время была собрана сильная команда специалистов и накоплен значительный опыт в области языков программирования и смежных технологий. Во-вторых, мы не можем сказать, что какой-либо из существующих языков на платформе Java полностью удовлетворяет нашим потребностям, и полагаем, основываясь на предварительных отзывах программистов всего мира, что наши коллеги в других компаниях испытывают похожие затруднения.

</cite>

</p>

<div id="citelegend">

Андрей Бреслав. Язык программирования Kotlin // Открытые системы. – 2011. – № 09

</div>

<p>

Kotlin 1.0 был выпущен 15 февраля 2016 года. Он считается первым официально стабильным релизом и начиная с этой версии, JetBrains взяла на себя обязательство по долгосрочной обратной совместимости.

</p>

<p>

В мае 2017 на Google I/O 2017 года компания Google объявила, что инструменты языка Kotlin, основанные на

JetBrains IDE, будут включены в Android Studio 3.0 – официальный инструмент разработки для ОС

Android.

</p>
<p>
Kotlin 1.2 был выпущен 28 ноября 2017 года. В релиз добавлена функция совместного использования кода между платформами JVM и JavaScript (мультиплатформенное программирование).

</p>
<p>
Kotlin 1.3 был выпущен 29 октября 2018 года, добавив поддержку сопрограмм для использования с асинхронным программированием.

</p>
<p>
На Google I/O 2019 было объявлено, что язык программирования Kotlin стал приоритетным в разработке под Android.

</p>
<p>
Kotlin 1.4 был выпущен в августе 2020 года, в том числе с некоторыми небольшими изменениями в поддержке платформ Apple (во взаимодействии Objective-C / Swift).

</p>
<p>
В ноябре 2020 года Андрей Бреслав объявил об уходе из JetBrains, руководство разработкой языка было передано Роману Елизарову.

</p>
<p>
Kotlin 1.5 был выпущен в мае 2021 года.

</p>
<p>
Kotlin 1.6 был выпущен в ноябре 2021 года.

</p>
<p>
Kotlin 1.7 был выпущен в июне 2022 года, включая альфа-версию нового компилятора Kotlin K2.

</p>
<p>
Kotlin 1.8 был выпущен в декабре 2022 года.

</p>
<p>
Kotlin 1.9 был выпущен в июле 2023 года.

</p>
<p>
Kotlin 2.0 был выпущен в мае 2024 года.

</p>
<h3>Синтаксис</h3>
<p>

Синтаксис языка преимущественно комбинирует наследство из двух языковых ветвей: Си/C++/Java и ML (по словам создателей, через Scala).

</p>

<p>

Из наиболее характерных элементов от первой ветви унаследованы блоки кода, обрамлённые фигурными скобками; а от второй – постфиксное указание типов переменных и параметров (сперва идентификатор, затем разделитель – двоеточие, и затем тип) и ключевые слова «fun» и «val». Точка с запятой как разделитель операторов необязательна (как в Scala, Groovy и JavaScript); в большинстве случаев перевода строки достаточно, чтобы компилятор понял, что выражение закончилось.

</p>

<p>

Кроме объектно-ориентированного подхода, Kotlin также поддерживает процедурный стиль с использованием функций. Как и в Си, C++ и D, точка входа в программу – функция main, принимающая массив параметров командной строки. Программы на Kotlin также поддерживают perl- и shell-стиль интерполяции строк (переменные, включённые в строку, заменяются на своё содержимое). Также поддерживается вывод типов.

</p>

<h3>Дальнейшее развитие</h3>

<p>

<!-- список в соответствии с заданием - основное задание -->

<ol id="list">

Компилятор

<!-- дополнительное задание - применение различных уровней css -->

>

<ol id="or" class="decimal" style="list-style-type: decimal">

Улучшить качество рекомендаций компилятора

Генерация методов JVM по умолчанию

Мультиплатформенность

<ol class="roman">

Публичный релиз Swift Export

Включенная по умолчанию многопоточная работа GC (Mark

+ Sweep)

Инструменты

<ol class="decimal">

Полноценный релиз IntelliJ IDEA K2

Разработка Design Tools API


```

        </li>
    </ol>
</p>

<h2>Лекция от создателя языка</h2>

<!-- video - дополнительное задание -->
<video width="100%" controls>
    <source src="img/breslav.mp4" type="video/mp4">
    Your browser does not support the video tag.
</video>

</div>

<div id="clear"> </div>

<!-- подвал - обязательный элемент сайта -->
<div id="footer">
    <h2>Контакты</h2>
    <p>
        <a href="https://github.com/maxi7665">GitHub</a><br>
        <a href="https://github.com/maxi7665/web-technologies">Исходный
код сайта</a>
    </p>
</div>
</div>
</body>

</html>

```

2. Файл “1.css”

```

body {
    background: #f1f1f1;
    color: #000;
    font-family: Arial, sans-serif;
    font-size: 14px;
    padding: 10px;
}

#header {
    background: #DCDCDC;
    width: 100%;
    height: 100px;
    display: flex;
    align-items: center;

```

```

}

#header h2 {
    text-align: left;
}

#sitename {
    margin-left: auto;
}

#pagename {
    margin-left: auto;
    margin-right: auto;
    font-size: 30;
}

#logo {
    height: 100%;
    width: 16%;
    text-align: left;
}

#logo img{
    height: 100%;
    width: auto;
}

#navigation {
    background: rgb(51,51,51);
    width: 100%;
    height: 50px;
}

#sidebar {
    background: #DCDCDC;
    float: left;
    width: 20%;
}

#content {
    background: #DCDCDC;
    float: right;
    width: 79%;
}

#content p {
    text-align: left;
    padding-left: 10px;
    padding-right: 10px;
}

```

```

#clear {
    clear: both;
}

#footer {
    /* дополнительное задание: применение градиента */
    background-image: linear-gradient(#333, #000000);
    width: 100%;
}

#footer h2 {
    color: #f2f2f2;
}

#footer a {
    color: #f2f2f2;
    text-decoration: none;
}

#footer p {
    padding: 3px;
}

#container {
    background: #AAA;
    margin: auto auto;
    text-align: center;
    width: 80%;
    padding: 10px;
}

#navigation a {
    float: left;
    display: block;
    color: #f2f2f2;
    text-align: center;
    padding: 14px 16px;
    text-decoration: none;
}

/* Change color on hover */
#navigation a:hover {
    background-color: #ddd;
    color: black;
}

#list{
    text-align: left;
}

```

```
#sitelegend{
    text-align: right;
    padding-right: 10px;
}

/* дополнительное задание - комбинатор "+" */
h3 + p {
    margin-top: 20px;
}
```

3. Файл “list.css”

```
/* дополнительное задание - связанные таблицы стилей */
ol.roman {
    list-style-type: upper-roman;
}

ol.decimal {
    list-style-type: decimal-leading-zero;
}
```

4. Файл “2.html”

```
<html>

<!-- шапка - обязательный элемент сайта -->

<head>
    <title>Язык Kotlin</title>
    <link rel="shortcut icon" type="image/x-icon" href="img/kotlin_logo.png" />
    <link rel="stylesheet" type="text/css" href="2.css">

    <meta charset="UTF-8">
    <!-- meta - дополнительное задание -->
    <meta name="description" content="Kotlin programming language">
    <meta name="keywords" content="Kotlin, Java, JVM, JDK">
    <meta name="author" content="Maxim Bystrov">

    <!-- основное задание - глобальные таблицы стилей -->
    <style>
        .table {
```

```

        border: 2px solid black;
        overflow-x: scroll;
        width: 300px;
        display: block;
        border-radius: 3%;
    }

    tr, th, td {
        border: 1px solid grey;
    }
</style>
</head>

<body>
    <div id="container" class="container">
        <header id="header">
            <div id="logo"></div>

            <h2 id="sitename">Язык Kotlin</h2>

            <h3 id="pagename">Документация</h3>

        </header>

        <!-- навигация между страницами - обязательный элемент сайта -->
        <nav id="navigation" class="container">
            <h2><a href="1.html">История</a>
                <a href="2.html">Документация</a>
                <a href="3.html">Источники</a>
            </h2>
        </nav>

        </p>

        <!-- боковая панель -->
        <aside id="sidebar">
            <h2>Новости</h2>
            <h3><a href="https://kotlinlang.org/docs/releases.html">Release
Notes</a></h3>
            <h3><a href="https://blog.jetbrains.com/kotlin/2024/11/kotlin-
roundup-kodee-s-top-picks/">Kotlin Roundup</a>
            </h3>
            <h3><a href="https://blog.jetbrains.com/kotlin/2024/10/ktor-3-
0/">Вышел Ktor 3.0</a></h3>
        </aside>

        <!-- основной контент -->
        <section id="content">

```

```

<article>
  <h2>Статус поддержки последних версий</h2>

  <!-- таблица - основное и дополнительное задание -->
  <table id="table" class="table" cellpadding="4" cellspacing="0">
    <colgroup>
      <col>
      <col span="3" style="background-color:yellow">
      <col style="background-color:green">
    </colgroup>
    <caption>Описание поддержки версий языка Kotlin</caption>
    <thead>
      <tr>
        <th>Версия</th>
        <th>1.7</th>
        <th>1.8</th>
        <th>1.9</th>
        <th>2.0</th>
      </tr>
    </thead>
    <tbody>
      <tr>
        <th>Обновления безопасности</th>
        <!-- основное задание - локальная таблица стилей -->
        <td colspan="3">Нет </td>
        <td>Да</td>
      </tr>
      <tr align="center">
        <th>Дата выпуска</th>
        <td>09 Jun 2022</td>
        <td>28 Dec 2022</td>
        <td>06 Jul 2023</td>
        <td>21 May 2024</td>
      </tr>
    </tbody>
  </table>
</article>
<hr>

<article>

```

<h2>Kotlin Multiplatform</h2>

<p>Поддержка мультиплатформенного программирования является одним из ключевых преимуществ Kotlin. Она сокращает время, затрачиваемое на написание и поддержку одного и того же кода для разных платформ,

сохраняя при этом гибкость и преимущества нативного программирования.</p>

```
<!-- набор картинок - основное задание -->
<picture id="pic1">
    <source srcset="img/kotlin_multiplatform_scheme.png"
type="image/png" />
    
</picture>

<h3>Примеры использования Kotlin Multiplatform</h3>

<!-- strong - дополнительное задание -->
<p><strong>Android и iOS приложения</strong></p>

<picture id="pic2">
    
</picture>

<!-- mark - дополнительное задание -->
<p>Использование <mark>одного кода</mark> на всех мобильных
платформах является одним из основных
вариантов
использования Kotlin Multiplatform. С помощью Kotlin
Multiplatform Mobile вы можете создавать
мультиплатформенные мобильные приложения, совместно
использующие один код и в Android, и в iOS,
например
бизнес-логику, возможности подключения и многое другое.</p>

<p><strong>Фулстек веб-приложения</strong></p>

<picture id="pic3">
    
</picture>

<p>Другой сценарий, когда совместное использование кода может
принести пользу, – это приложение, в
котором
логика может быть повторно использована как на стороне
сервера, так и на стороне клиента,
запущенного в
браузере. С этим Kotlin Multiplatform тоже прекрасно
справляется.</p>

<p><strong>Мультиплатформенные библиотеки</strong></p>
```

```

        <picture id="pic3">
            
        </picture>

        <!-- abbr - дополнительное задание -->
        <p>Kotlin Multiplatform также отлично подойдет для авторов
библиотек. Вы можете создать
            мультиплатформенную
            библиотеку с общим кодом и ее платформенными реализациями для
<abbr>JVM</abbr>, <abbr>JS</abbr> и
            собственных платформ. После
            публикации мультиплатформенная библиотека может
использоваться в других кроссплатформенных проектах
            в
            качестве зависимости.</p>

        <p><strong>Общий код для мобильных и веб-приложений</strong></p>

        <picture id="pic3">
            
        </picture>

        <p>Еще одним популярным примером использования Kotlin
Multiplatform является совместное использование
            одного и того же кода в Android, iOS и веб-приложениях. Это
сокращает объем бизнес-логики,
            написанной
            фронтенд-разработчиками, и помогает более эффективно внедрять
продукты, сокращая затраты на
            разработку и
            тестирование.
        </p>
    </article>
    <hr>

    <h2>Определение имени пакета и импорт</h2>
    <p>Имя пакета указывается в начале исходного файла, так же как и в
Java.</p>
    <pre>
        <!-- здесь и далее code - дополнительное задание -->
        <code class="language-kotlin hljs" data-
highlighted="yes">
            package my.demo

            import java.util.*

            // ...</code>

```

```
</pre>
<p>Но в отличие от Java, нет необходимости, чтобы структура пакетов совпадала со структурой папок:
    исходные файлы могут располагаться в произвольном месте на диске.</p>
```

<hr>

<h2>Точка входа в программу</h2>

<p>В Kotlin точкой входа в программу является функция main.</p>

```
<pre>
    <code class="language-kotlin hljs" data-highlighted="yes">
fun main() {
    println("Hello world!")
}</code>
```

```
</pre>
```

<p>Другая форма main может принимать массив строк String.</p>

```
<pre>
    <code class="language-kotlin hljs" data-highlighted="yes">
fun main(args: Array<String>) {
    println(args.contentToString())
}</code>
```

```
</pre>
```

<hr>

<h2>Вывод в стандартный поток</h2>

<p>print выводит свой аргумент в стандартный поток вывода.</p>

```
<pre>
    <code class="language-kotlin hljs" data-highlighted="yes">
print("Hello")
print("world!")</code>
```

```
</pre>
```

<p>println выводит свой аргумент и добавляет перевод строки, так что следующее, что вы выведете, появится на следующей строке.</p>

```
<pre>
    <code class="language-kotlin hljs" data-highlighted="yes">
println("Hello world!")
println(42)</code>
```

```
</pre>
```

<hr>

<h2>Функции</h2>

<p>Функция принимает два аргумента Int и возвращает Int.</p>

```
<pre>
    <code class="language-kotlin hljs" data-highlighted="yes">
```

```

fun sum(a: Int, b: Int): Int {
    return a + b
}

```

В качестве тела функции может выступать выражение. Тогда тип возвращаемого значения определяется автоматически.

```

fun sum(a: Int, b: Int) = a + b

```

Функция, не возвращающая никакого значения (void в Java).

```

fun printSum(a: Int, b: Int): Unit {
    println("сумма $a и $b равна ${a + b}")
}

```

Тип возвращаемого значения Unit может быть опущен.

```

fun printSum(a: Int, b: Int) {
    println("сумма $a и $b равна ${a + b}")
}

```

Переменные

Неизменяемые (только для чтения) локальные переменные определяются с помощью ключевого слова `val`.

Присвоить им значение можно только один раз.

```

val a: Int = 1 // Инициализация при объявлении
val b = 1      // Тип `Int` определен автоматически
val c: Int     // Указывать тип обязательно,
// если переменная не инициализирована сразу
c = 1          // Последующее присвоение

```

Изменяемые переменные объявляются с помощью ключевого слова `var`.

```

var x = 5 // Тип `Int` определен автоматически
x += 1

```

```

        </pre>
        <p>Вы можете объявлять глобальные переменные.</p>
        <pre>
            <code class="language-kotlin hljs" data-
highlighted="yes">
                val PI = 3.14
                var x = 0

                fun incrementX() {
                    x += 1
                }</code>
            </pre>
        </section>

        <div id="clear"> </div>

        <!-- подвал - обязательный элемент сайта -->
        <footer id="footer">
            <h2>Контакты</h2>
            <p>
                <a href="https://github.com/maxi7665">GitHub</a>
            </p>
            <a href="https://github.com/maxi7665/web-technologies">Исходный код
сайта</a>
            </p>
        </footer>
    </div>
</body>

</html>

```

5. Файл “2.css”

```

body {
    background: #f1f1f1;
    color: #000;
    font-family: Arial, sans-serif;
    font-size: 14px;
    padding: 10px;
}

/* body {
    font-family: Arial;
    padding: 10px;
    background: #f1f1f1;
} */

```

```

#header {
    background: #DCDCDC;
    width: 100%;
    height: 100px;
    display: flex;
    align-items: center;
}

#header h2 {
    text-align: left;
    /* padding: 10px; */
}

#sitename {
    margin-left: auto;
}

#pagename{
    margin-left: auto;
    margin-right: auto;
    font-size: 30;
}

#logo {
    height: 100%;
    width: 16%;
    text-align: left;
}

#logo img{
    height: 100%;
    width: auto;
}

#navigation {
    background: #333;
    width: 100%;
    height: 50px;
}

#sidebar {
    background: #DCDCDC;
    float: left;
    width: 20%;
}

#content {
    background: #DCDCDC;

```

```

        float: right;
        width: 79%;
        /* height: 100%; */
    }

    #content p {
        text-align: left;
        padding-left: 10px;
        padding-right: 10px;
    }

    #clear {
        clear: both;
    }

    #footer {
        background: #333;
        width: 100%;
        /* height: 40px; */
    }

    #footer h2 {
        color: #f2f2f2;
    }

    #footer a {
        color: #f2f2f2;
        text-decoration: none;
    }

    #footer p {
        padding: 3px;
    }

    #container {
        background: #AAA;
        margin: auto auto;
        text-align: center;
        width: 80%;
        /* height: 800px; */
        padding: 10px;
    }

    #navigation a {
        float: left;
        display: block;
        color: #f2f2f2;
        text-align: center;
        padding: 14px 16px;
        text-decoration: none;
    }

```

```

}

/* Change color on hover */
#navigation a:hover {
    background-color: #ddd;
    color: black;
}

#list{
    text-align: left;
}

#table {
    margin: 10px;
}

#content picture img {
    width: 80%;
}

#content pre{
    text-align: left;
}

abbr
{
    font-style: italic;
    color: chocolate;
}

```

6. Файл “3.html”

```

<html>

<head>
    <title>Язык Kotlin</title>
    <link rel="shortcut icon" type="image/x-icon" href="img/kotlin_logo.png" />
    <link rel="stylesheet" type="text/css" href="3.css">

    <meta charset="UTF-8">
    <!-- meta - дополнительное задание -->
    <meta name="description" content="Kotlin programming language">
    <meta name="keywords" content="Kotlin, Java, JVM, JDK">
    <meta name="author" content="Maxim Bystrov">
</head>

```



```

<body>
  <div id="container" class="container">

    <!-- шапка - обязательный элемент сайта -->
    <div id="header">
      <div id="logo"></div>

      <h2 id="sitename">Язык Kotlin</h2>

      <h3 id="pagename">Источники</h3>

    </div>

    <!-- навигация между страницами - обязательный элемент сайта -->
    <div id="navigation" class="container">
      <h2><a href="1.html">История</a>
        <a href="2.html">Документация</a>
        <a href="3.html">Источники</a>
      </h2>
    </div>

  </p>

  <!-- боковая панель -->
  <div id="sidebar">
    <h2>Новости</h2>
    <h3><a href="https://kotlinlang.org/docs/releases.html">Release
Notes</a></h3>
    <h3><a href="https://blog.jetbrains.com/kotlin/2024/11/kotlin-
roundup-kodee-s-top-picks/">Kotlin Roundup</a></h3>
    <h3><a href="https://blog.jetbrains.com/kotlin/2024/10/ktor-3-
0/">Вышел Ktor 3.0</a></h3>
  </div>

  <!-- основной контент -->
  <div id="content">
    <p><a href="https://kotlinlang.org/docs/home.html">Документация
Kotlin</a></p>
    <p><a href="https://blog.jetbrains.com/kotlin/">Kotlin Blog</a></p>
    <p><a href="https://developer.mozilla.org/en-US/">MDN Web
Docs</a></p>
    <p><a href="https://www.w3schools.com/html/default.asp">W3
Schools</a></p>
    <p><a href="https://www.osp.ru/os/2011/09/13011550">Открытые системы.
Язык программирования Kotlin</a></p>
  </div>

  <div id="clear"> </div>

```

```

    <!-- подвал - обязательный элемент сайта -->
    <div id="footer">
        <h2>Контакты</h2>
        <p>
            <a href="https://github.com/maxi7665">GitHub</a>
        </p>
        <a href="https://github.com/maxi7665/web-technologies">Исходный код
сайта</a>
        </p>
    </div>
</div>
</body>

</html>

```

7. Файл “3.css”

```

body {
    background: #f1f1f1;
    color: #000;
    font-family: Arial, sans-serif;
    font-size: 14px;
    padding: 10px;
}

/* body {
    font-family: Arial;
    padding: 10px;
    background: #f1f1f1;
} */

#header {
    background: #DCDCDC;
    width: 100%;
    height: 100px;
    display: flex;
    align-items: center;
}

#header h2 {
    text-align: left;
    /* padding: 10px; */
}

```

```

}

#sitename {
    margin-left: auto;
}

#pagename{
    margin-left: auto;
    margin-right: auto;
    font-size: 30;
}

#logo {
    height: 100%;
    width: 16%;
    text-align: left;
}

#logo img{
    height: 100%;
    width: auto;
}

#navigation {
    background: #333;
    width: 100%;
    height: 50px;
}

#sidebar {
    background: #DCDCDC;
    float: left;
    width: 20%;
    /* height: 100%; */
}

#content {
    background: #DCDCDC;
    float: right;
    width: 79%;
    /* height: 100%; */
}

#content p {
    text-align: left;
    padding-left: 10px;
    padding-right: 10px;
}

#clear {

```

```

        clear: both;
    }

    #footer {
        background: #333;
        width: 100%;
        /* height: 40px; */
    }

    #footer h2 {
        color: #f2f2f2;
    }

    #footer a {
        color: #f2f2f2;
        text-decoration: none;
    }

    #footer p {
        padding: 3px;
    }

    #container {
        background: #AAA;
        margin: auto auto;
        text-align: center;
        width: 80%;
        /* height: 800px; */
        padding: 10px;
    }

    #navigation a {
        float: left;
        display: block;
        color: #f2f2f2;
        text-align: center;
        padding: 14px 16px;
        text-decoration: none;
    }

    /* Change color on hover */
    #navigation a:hover {
        background-color: #ddd;
        color: black;
    }

    #list{
        text-align: left;
    }

```

```
#table {  
    margin: 10px;  
}  
  
#content picture img {  
    width: 80%;  
}  
  
#content pre{  
    text-align: left;  
}  
  
abbr  
{  
    font-style: italic;  
    color: chocolate;  
}  
  
#pagename::first-letter  
{  
    font-size: 35px;  
}
```

Вывод

В ходе выполнения второй лабораторной работы были созданы и применены каскадные таблицы стилей (CSS).

Выполнены пункты обязательного и дополнительного задания.

Применены связанные, глобальные и локальные таблицы стилей, показан принцип применения стилей.

Приведены скриншоты и исходный код разработанного сайта.