МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное
образовательное учреждение высшего образования
«ДАГЕСТАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
Факультет информатики и информационных технологий

# Интернет программирование

*Учебно-методическое пособие*

Москва, 2025

**УДК 004.42**
**ББК 32.973.4я73**
**Б22**

Баммаева Г.А.
Мусаева У.А.
**Б22 ИНТЕРНЕТ ПРОГРАММИРОВАНИЕ: УЧЕБНОЕ МЕТОДИЧЕСКОЕ ПОСОБИЕ.** – *М.: Издательство «Перо», 2025. – 90 с.*

**ISBN 978-5-00270-500-9**

Учебное пособие содержит 12 лабораторных работ и позволяет освоить язык разметки гипертекста HTML, каскадную таблицу стилей CSS в их классическом представлении, а также овладеть технологией web программирования на языке JavaScript. Каждая лабораторная работа включает теоретические сведения, практические примеры в среде HTML. Пособие предназначено для бакалавров высших учебных заведений, изучающих основы интернет программирования и объектно-ориентированное программирование на языке JavaScript, а также тех, кто хочет самостоятельно освоить программирование на HTML, CSS, JavaScript.

**ISBN 978-5-00270-500-9**




© Баммаева Г.А., Мусаева У.А., 2025

# ОГЛАВЛЕНИЕ

**Введение** ..... **4**

**Раздел I. Введение в язык HTML** ..... **6**

*Лабораторная работа №1.* ..... *10*

*Лабораторная работа №2.* ..... *14*

*Лабораторная работа №3.* ..... *17*

*Лабораторная работа №4* ..... *19*

*Лабораторная работа №5* ..... *25*

*Лабораторная работа №6* ..... *31*

*Лабораторная работа №7* ..... *37*

*Лабораторная работа №8* ..... *43*

*Лабораторная работа №9* ..... *50*

**Раздел II. Введение в CSS** ..... **58**

*Лабораторная работа № 1* ..... *58*

*Лабораторная работа №2* ..... *64*

**Раздел III. Введение в JavaScript** ..... **75**

*Лабораторная работа №1* ..... *79*

**Список используемых источников** ..... **88**

# ВВЕДЕНИЕ

Современное общество вступило в информационную эпоху своего исторического развития. На планете миллиарды людей связаны единым каналом общения и передачи информации. Человек получил практически неограниченный доступ к библиотекам, книгам - знаниям всех эпох и культур. Интернет - это потребность жизни XXI в, такая же необходимая, как телефон, телевизор, автомобиль. В мире с каждым годом объем передаваемой информации утраивается. Не случайно XXI в. называют эрой информационного общества.

Интернет, самая разветвленная сеть, к которой подключены самолеты и автомобили, дома и бытовые и приборы, в промышленности царит автоматическое производство, контроль за качеством и организацией производства стало возможным из любой точки земного шара. Каждый год объем передаваемой информации удваивается. Не случайно XXI в. называют эрой информационного общества.

Без опережающего развития собственной телекоммуникационной отрасли и интеграции в мировую систему инфокоммуникаций Россия не может рассчитывать на подъем национальной экономики. Базой для обеспечения открытости органов государственной власти, развития дистанционного образования, телемедицины, электронной торговли являются современные телекоммуникационные системы.

Интернет - всемирная система объединённых компьютерных сетей для хранения и передачи информации. Часто упоминается как Всемирная сеть и Глобальная сеть, а также просто Сеть. Построена на базе стека протоколов TCP/IP. На основе Интернета работает Всемирная паутина (World Wide Web, WWW) и множество других систем передачи данных.

По данным на 2025 год, число интернет-пользователей во всем мире составило 6 миллиардов человек, что составляет примерно 74% от мирового населения. Это увеличение по сравнению с 5,5 миллиардами в предыдущем году.

HTML (HyperText Markup Language - «язык разметки гипертекста») - стандартный язык разметки документов во Всемирной паутине. Большинство веб-страниц созданы при помощи языка HTML. Язык HTML интерпретируется браузером и отображается в виде документа, в удобной для человека форме.

HTML является приложением SGML (стандартного обобщённого языка разметки) и соответствует международному стандарту ISO 8879.

HTML создавался как язык для обмена научной и технической документацией, пригодный для использования людьми, не являющимися специалистами в области вёрстки. Для этого он представляет небольшой (сравнительно) набор структурных и семантических элементов - тегов. С помощью HTML можно легко создать относительно простой, но красиво оформленный документ. Изначально язык HTML был задуман и создан как средство структурирования и форматирования документов без их привязки к средствам воспроизведения (отображения). В идеале, текст с разметкой HTML должен единообразно воспроизводиться на различном оборудовании (монитор ПК, экран органайзера, ограниченный по размерам экран мобильного телефона, медиапроектор). Однако современное применение HTML очень далеко от его изначальной задачи. Со временем основная идея платформонезависимости языка HTML стала жертвой коммерциализации WWW и потребностей в мультимедийном и графическом оформлении.

Методическое пособие содержит 12 лабораторных работ и позволяет освоить язык разметки гипертекста HTML, каскадную таблицу стилей CSS,а также овладеть технологией интернет программирования на языке JavaScript. Каждая лабораторная работа включает теоретические сведения, и практические примеры, работающих в среде HTML.

Пособие предназначено для бакалавров высших учебных заведений, изучающих основы интернет и объектно-ориентированного программирований на языке JavaScript, а также тех, кто хочет самостоятельно освоить программирование на HTML, CSS, JavaScript.

# РАЗДЕЛ I.
# ВВЕДЕНИЕ В ЯЗЫК HTML

## Теоретическая справка

HTML (Hyper Text Markup Language) представляет собой язык разметки гипертекста. Из названия становится понятно, что предназначение языка заключается в том, чтобы указывать браузеру, как разместить элементы гипертекста на странице при ее отображении на экране с помощью специальных кодов, легко отделяемых от смыслового содержания документа. Гипертекст представляет собой совокупность документов, содержащих текстовую, аудио и видео информацию, связанных между собой взаимными ссылками в единый текст.

## Синтаксис HTML

HTML - документ представляет собой текстовый файл, который может быть создан в произвольном текстовом редакторе, однако существуют также и специальные HTML-редакторы. Он содержит собственно сам текст, который должен быть отражен в окне браузера, и команды разметки -HTML-теги, которые задают элементы документа, определяя его внешний вид и функциональность. Теги записываются в угловые скобки (<>) и состоят из имени элемента, за которым следуют его атрибуты.

Пример:

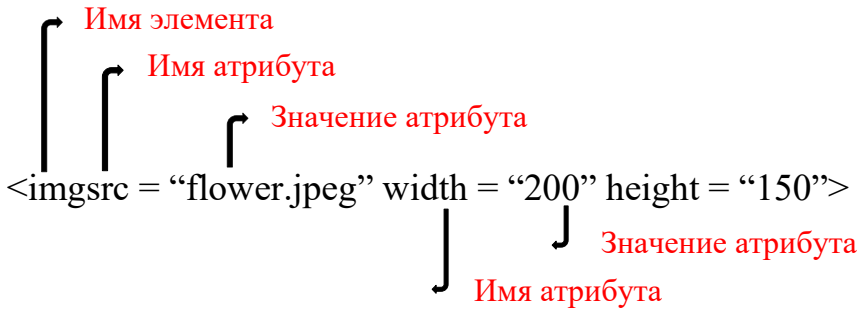
```
<b>Этот текст будет отображаться жирным</b>
<i>Этот текст будет выделен курсивом</i>
```

В данном примере теги и  делают текст жирным и курсивом соответственно.

Большинство элементов (которые называются также контейнерами) состоят из трех компонентов: начальный тег, содержимое и конечный тег. В начальном теге в угловых скобках указывается имя элемента и список атрибутов, в конечном - только имя, предваряемое символом слэш (/). Содержи-

мое элемента располагается между начальным и конечными тегами и интерпретируется браузером согласно правилам, определенным в спецификации стандарта HTML. Атрибутами называются свойства элементов.

Пример:




В примере приведен тег, не имеющий содержимого и потому не требующий конечного тега. Встретив такой тег в HTML-документе, браузер интерпретирует его, выведя изображение из файла flower.jpeg шириной в 200 и высотой в 150 пикселов.

Для полноты картины приведем и пример элемента-контейнера, имеющего содержимое и, соответственно, требующего наличия конечного тега.

Пример:

```
<palign = “right”>
…содержимое элемента…
</p>
```

После конечного тега можно объявлять новый абзац.

## Структура HTML

Структура HTML-документа состоит из трех пар тегов: html, head, body. Теги <html></html>являются контейнером для всех остальных, так как в них размещаются все остальные. Таким образом любой HTML-документ должен начинаться с тега <html> и заканчиваться тегом </html>. Сам документ условно разделен на две части заголовок документа и тело документа. Заголовок заключается в теги <head></head>, а тело в <body></body>. Заголовок содержит служебную информацию и не влияет на внешний вид документа.

Единственный обязательный элемент заголовка документа - это теги <title></title>. Они нужны, чтобы дать документу название, которое отражается в заголовке окна браузера.

Все, что отображается на web-странице находится в тегах <body></body>. Это текст, картинки и исполняющиеся скрипты, а также теги для оформления всего этого. Обязательных параметров у тега <body> нет.

Пример:

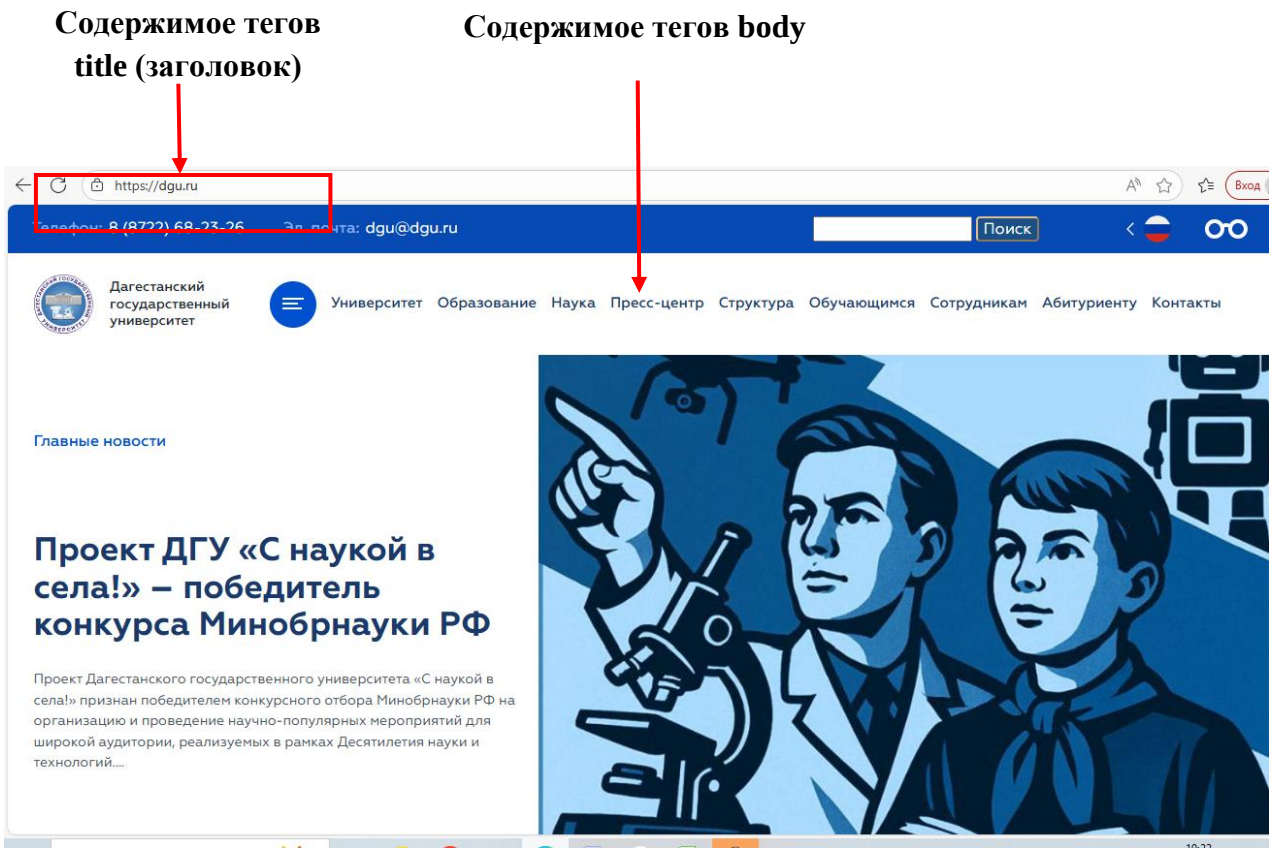



**Рисунок 1. Сайт Дагестанского государственного университета**

Код html-документа на примере сайта Дагестанского государственного университета приведен на рисунке 2.

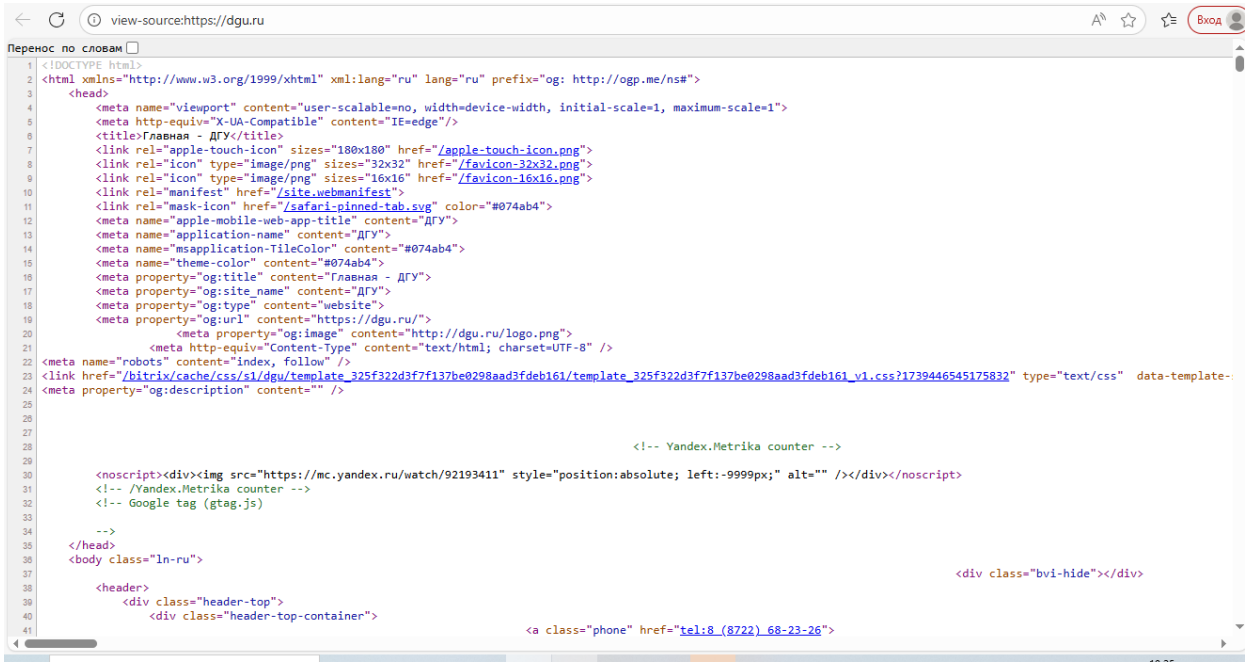



**Рисунок 2. Пример html-кода сайта ДГУ**

# Лабораторная работа №1.

## Создание первой HTML-страницы

Создать HTML-страницу можно в любом текстовом редакторе, даже Microsoft Word, главное сохранить созданный файл с расширением html или htm.

В ходе выполнения лабораторных работ, будет использован текстовый редактор **Visual Studio Code** (рисунок 1.1).

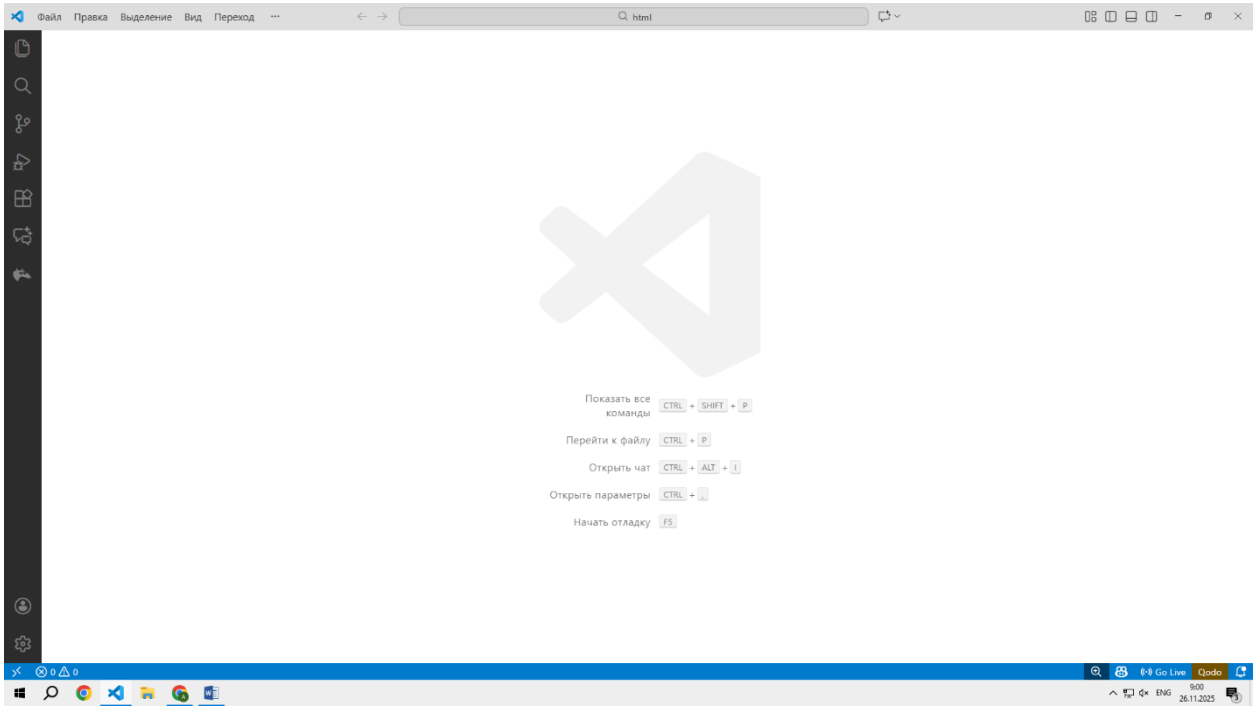



**Рисунок 1.1. Окно текстового редактора Visual Studio Code**

Перед началом работы необходимо проверить правильность настроек приложения. Для этого выберите пункт главного меню «Опции» и в контекстном меню выберите пункт «Настройки». В открывшемся окне, перейдите на вкладку «Новый документ» (рисунок 1.2).

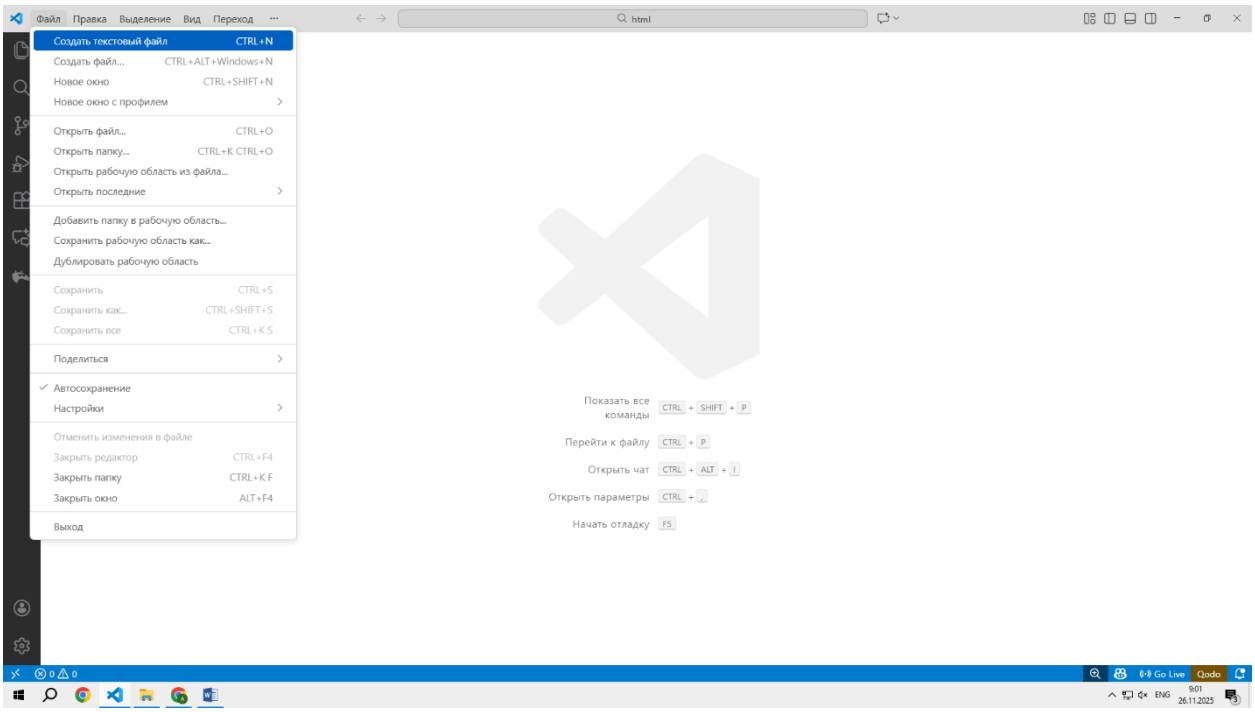



**Рисунок 1.2. Вкладка «Создать текстовый файл» пункта меню «Файл»**

Убедитесь в том, что в данном окне отмечены пункты «UTF-8» и «Применить к открытым ANSI файлам».

Далее выберите пункт главного меню «Кодировки» и убедитесь, что в контекстном меню установлена галочка на пункте «Кодировать в UTF-8 без BOM» (рисунок 1.3).

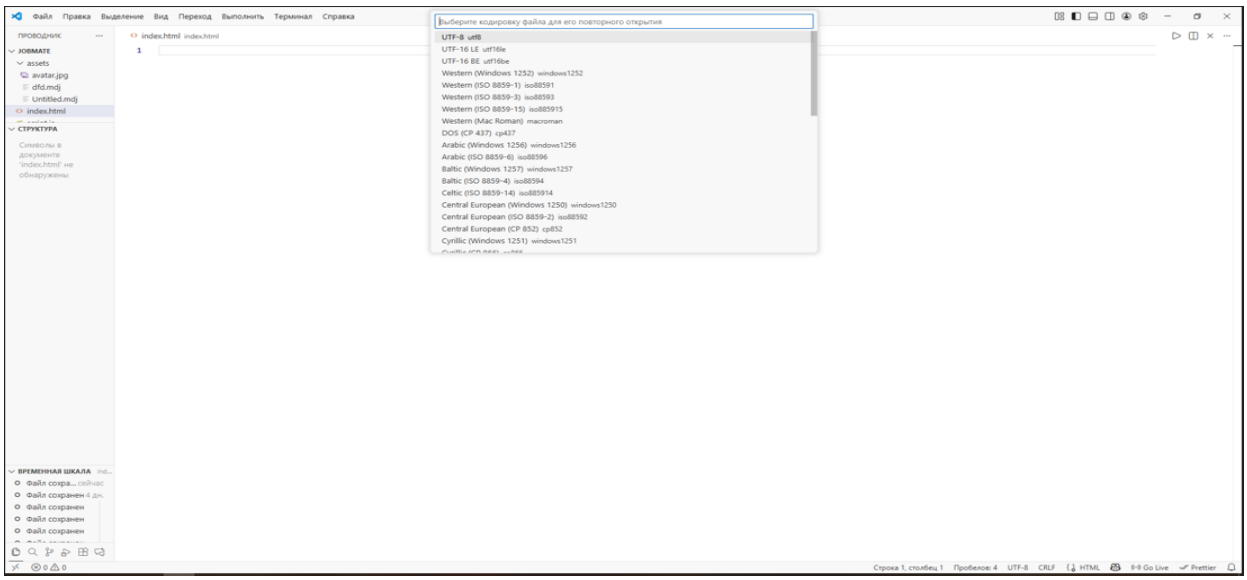


**Рисунок 1.3. Контекстное меню пункта главного меню «Кодировки»**

После выполнения вышеперечисленных действий можно приступать к

работе.

Для начала составим HTML-документ с простейшей структурой (рисунок 1.4).

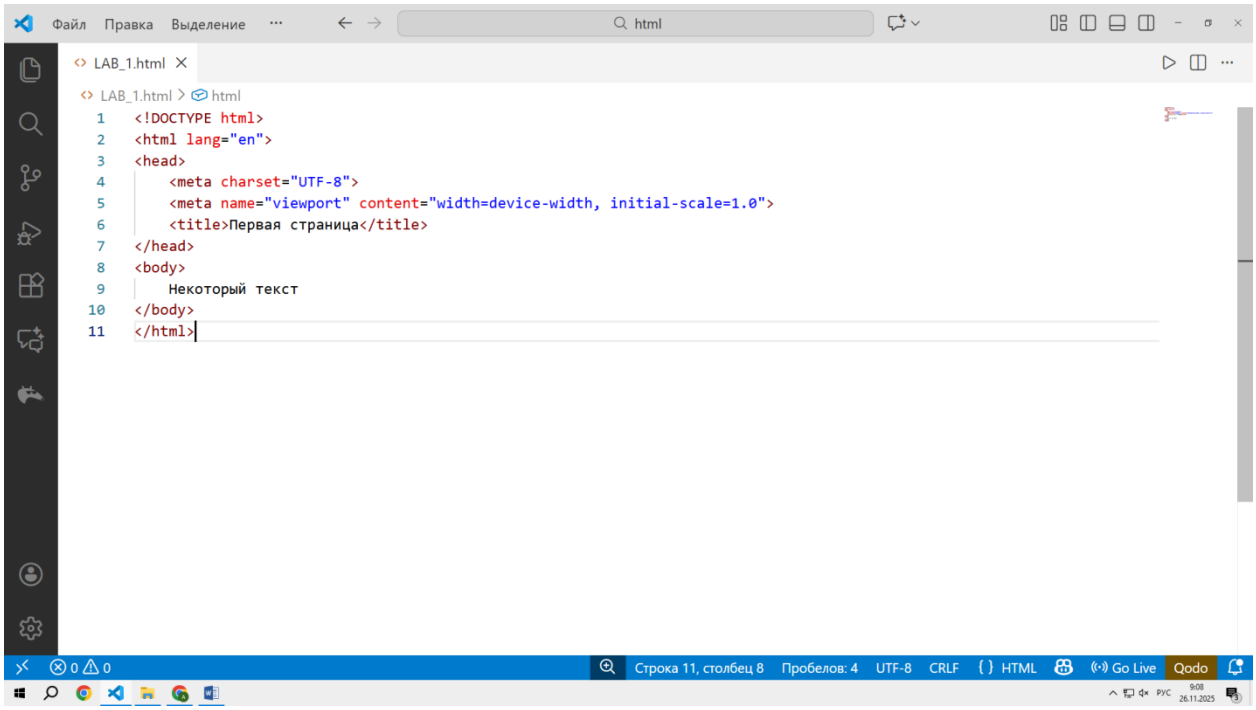



**Рисунок 1.4. Структура HTML-документа**

Как ранее упоминалось во вводной части, то, что должно отображаться визуально в теле документа, нужно прописать между открывающим и закрывающим тегами <body></body>. Чтобы поместить заголовок страницы на вкладку браузера нужно в блок документа <head></head> добавить элемент <title></title>, в котором и нужно записать название страницы. Данный элемент служит для описания страниц и используется поисковыми системами.

Созданный файл необходимо сохранить с расширением «*.html». В окне браузера он будет выглядеть, как на рисунке 1.5.

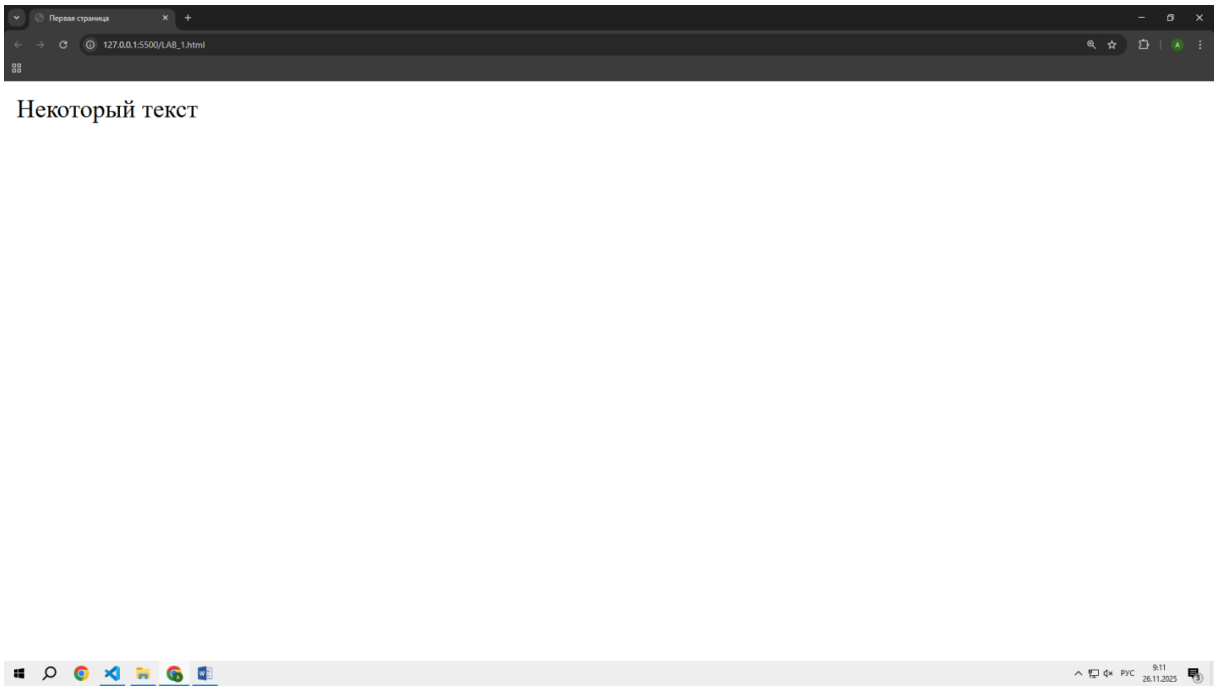



**Рисунок 1.5. Html-страница в окне браузера**

# Лабораторная работа №2.

## Параграфы в HTML

Во вводной части был приведен пример тега , как элемента-контейнера, имеющего содержимое и требующего конечный тег .

Данный тег создает отдельные параграфы текста (рисунок 2.1.).

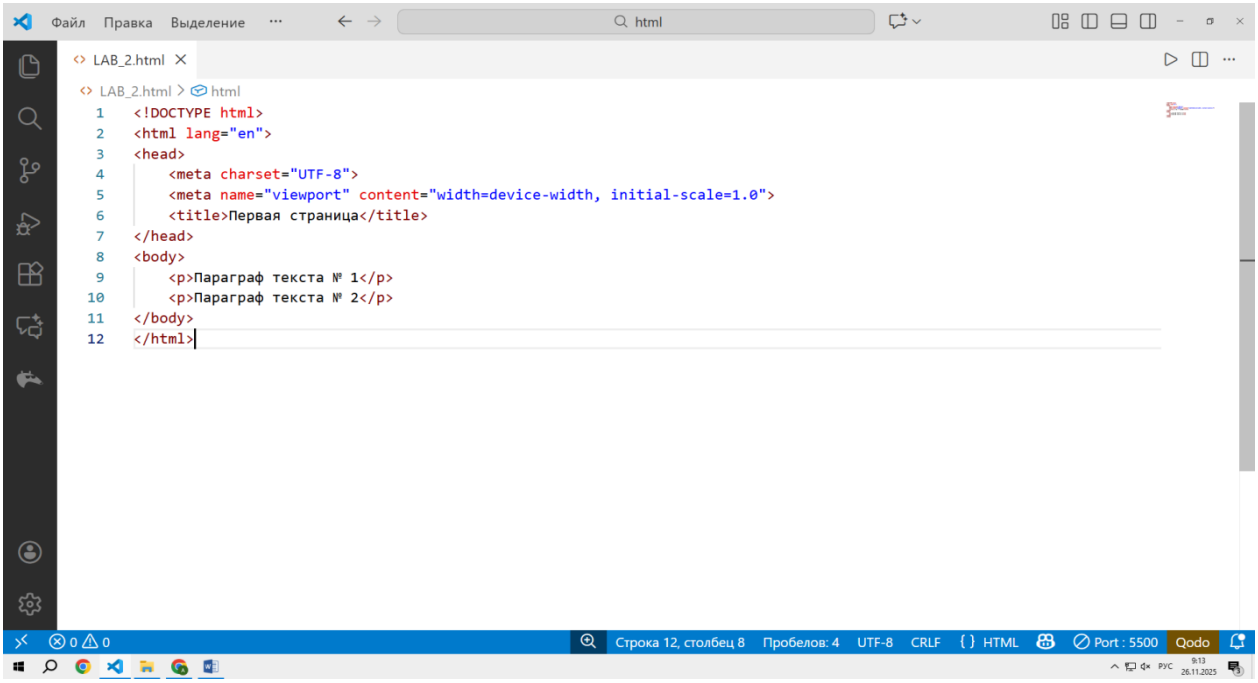



**Рисунок 2.1. Структура html-документа с использованием тега **

В окне браузера данная страница будет выглядеть как на рисунке 2.2.

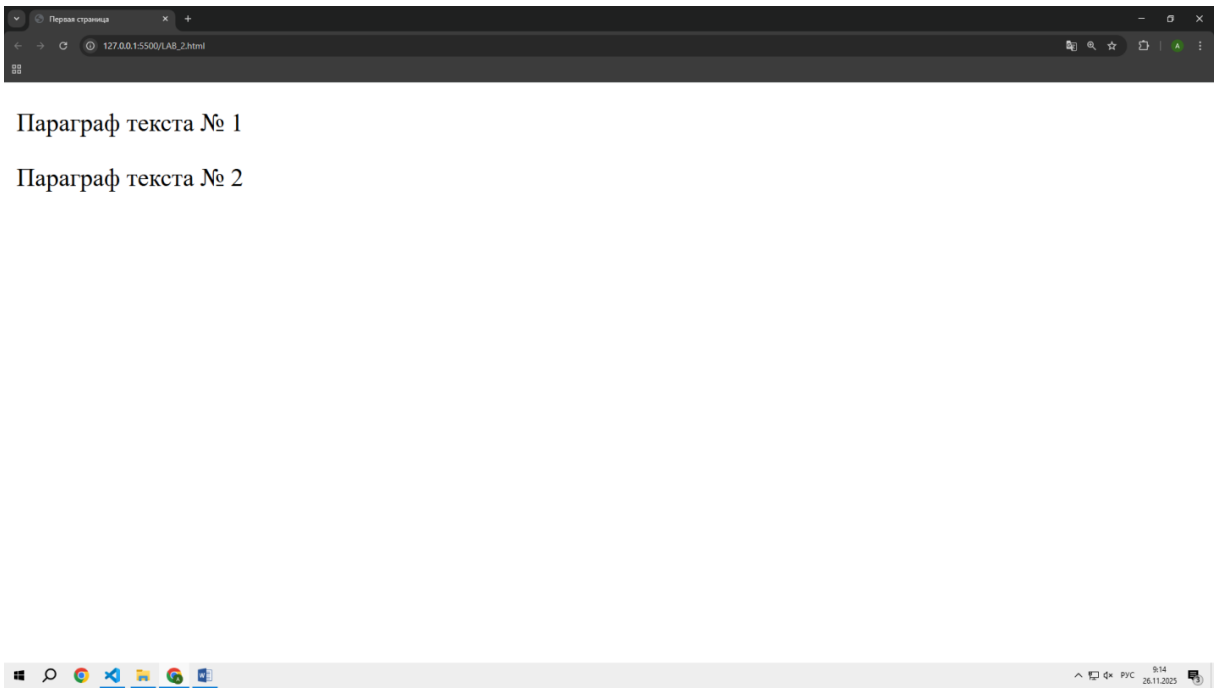



**Рисунок 2.2. Html-страница в окне браузера**

Как видно из рисунка 2.2 тег <p>создает автоматический перенос текста на новую строку. Всё записанное нами в блок <body>без дополнительных тегов в окне браузера отобразилось бы сплошным текстом, даже если бы мы его отформатировали привычными для нас способами с использованием встроенных инструментов текстового редактора, так как HTML не признает никакого дополнительного форматирования текста, кроме как с помощью тегов.

Однако для переноса строки существует также и специальный тег <br>. Приведем пример его использования. Для этого добавим в наш html-файл ещё одну строку (рисунок 2.3).

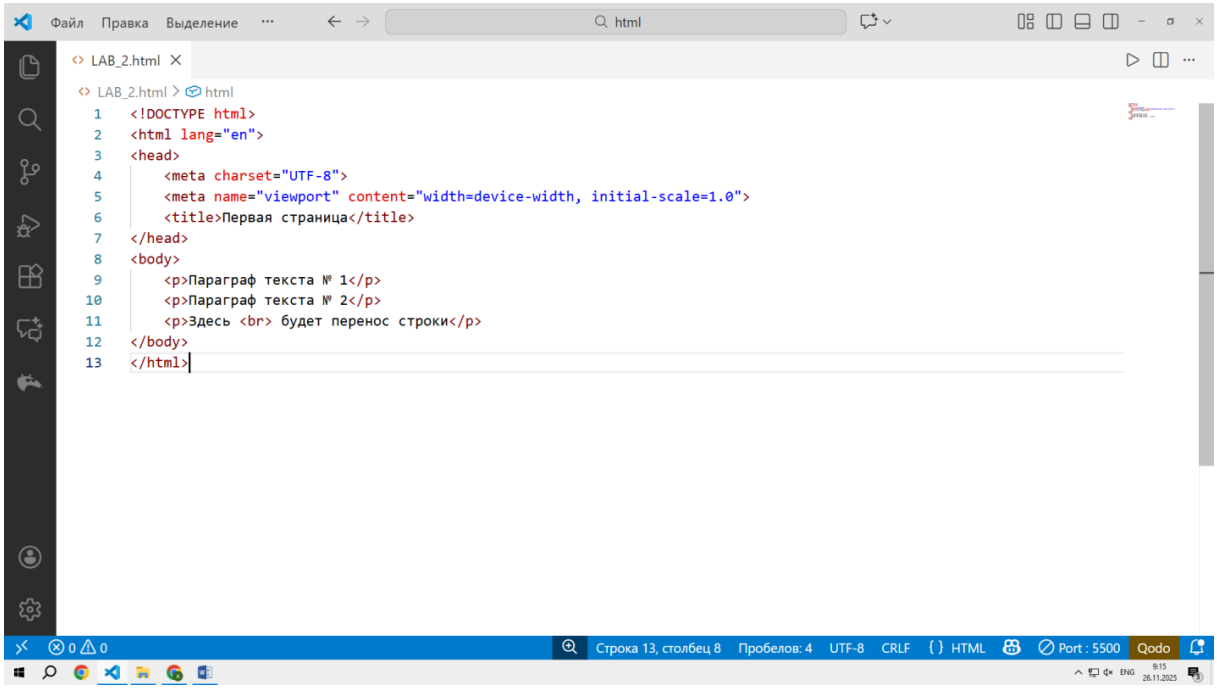



**Рисунок 2.3. Структура html-страницы с использованием тега для переноса строки <br>**

В окне браузера данная страница будет выглядеть как на рисунке 2.4.

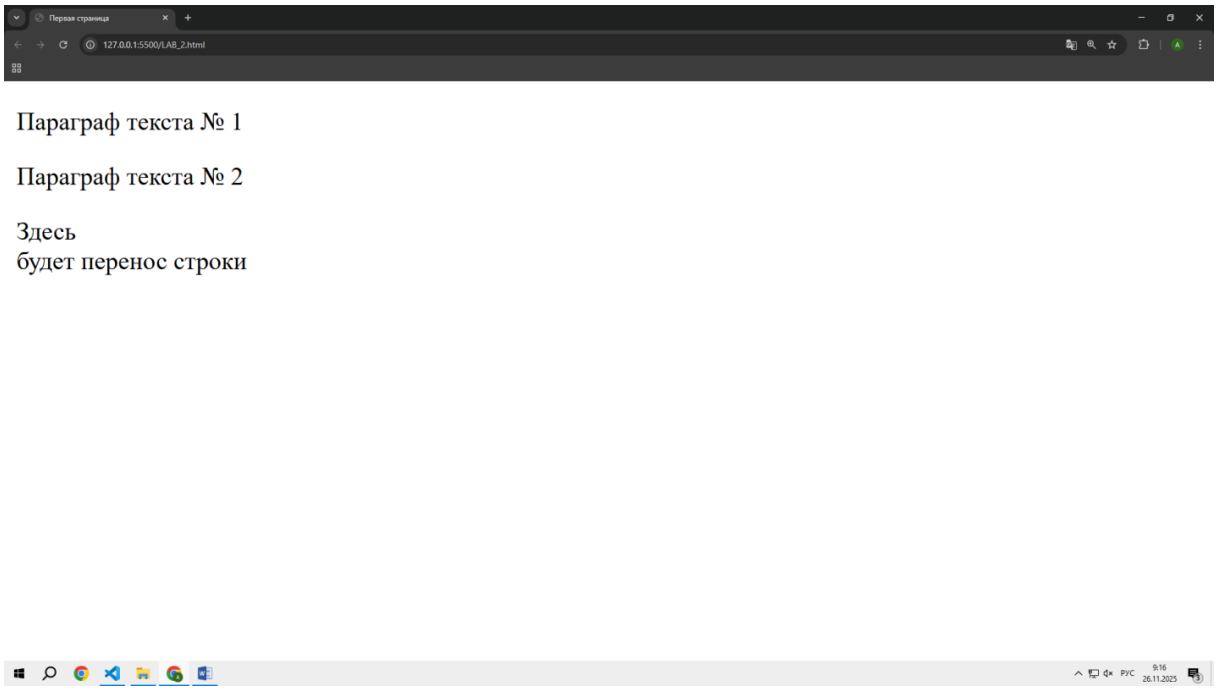



**Рисунок 2.4. Html-страница в окне браузера**

Соответственно, если мы уберем тег <br>, то у нас будет простой параграф текста, без переноса. Этот тег можно назвать «пустым» html-элементом, так как он не имеет содержимого и не требует конечного закрывающего тега. При этом тег <br>для переноса строки используется редко, т.к. для этого существуют и другие специальные теги, которые сохраняют переносы. Но <br> можно использовать, чтобы создавать пустые строки, отступ в тексте после других тегов.

# Лабораторная работа №3.

## Форматирование текста в HTML

В HTML используется ряд элементов, предназначенных для определения стиля представления текста. Рассмотрим некоторые из них.

*Таблица 1*

*Используемые теги в Html и их значения*

| Название тега | Стиль представления текста |
|---|---|
| <br> | Выделение важных фрагментов текста жирным шрифтом |
| <big></big> | Выделение текста на странице крупным шрифтом |
|  | Выделение текста на странице курсивом |
| <small></small> | Выделение текста на странице мелким шрифтом |
| <sub></sub> | Подстрочный текст (нижний индекс) |
| <sup></sup> | Надстрочный текст (верхний индекс) |
| <ins></ins> | Вставленный текст |
| <del></del> | Удаленный (перечеркнутый) текст |
| <ins></ins><br><u></u> | Оба тега используются для подчеркивания текста, но предпочтительнее использовать первый |
| <font></font> | Указывает параметры шрифта:<br>face- название шрифта;<br>size- размер шрифта;<br>color- цвет шрифта |
|  | Служит для выравнивания всех элементов внутри него по центру окна браузера |
| <address></address> | Применяется для указания адреса (отображается курсивом) |
| <abbr></abbr><br><acronym></acronym> | Используются для указания на то, что текст является аббревиатурой (последний тег поддерживается всеми браузерами). По умолчанию текст подчеркивается пунктиром, а при наведении на него курсора появляется всплывающая подсказка с расшифровкой. Это достигается за счет добавления в тег параметра title |
| <hr></hr> | Рисует горизонтальную линию, отделенную сверху и снизу пустыми строками |

Приведем пример кода с использованием всех вышеперечисленных тегов (рисунок 3.1).

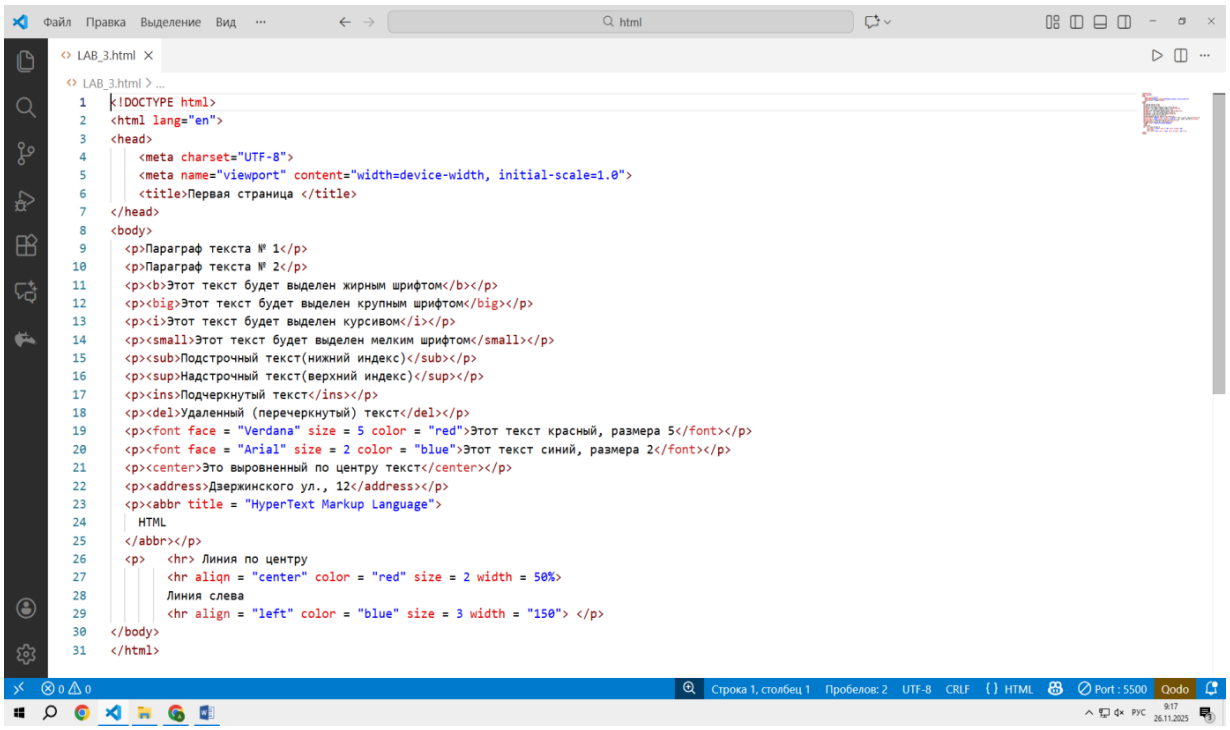



**Рисунок 3.1. HTML-код с использованием тегов для форматирования текста**

В окне браузера эта страница будет выглядеть как на рисунке 3.2.

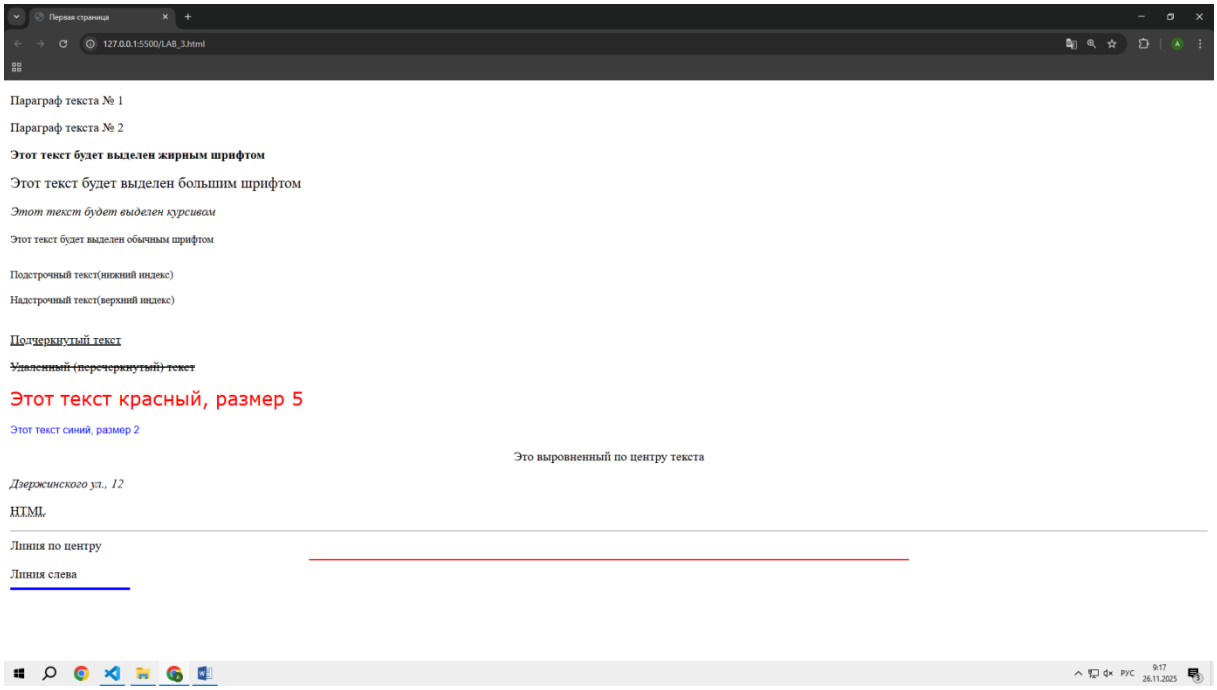



**Рисунок 3.2. Пример html-страницы в окне браузера**

# Лабораторная работа №4

## Заголовки, линии, комментарии, символы в HTML

### 4.1. Заголовки

Важность заголовков на странице определяется тем, что обычно заголовки первое и, часто, единственное, на что посетители сайта обращают внимание. Поэтому неправильное использование заголовков может привести к потере посетителей. В первую очередь заголовки должны кратко и точно описывать содержимое того, что они озаглавливают. Наиболее важная информация должна быть под заголовками большего размера, соответственно, наименее важная - под заголовками меньшего размера.

В Htmlиспользуется шесть уровней заголовков, организованных по уровню важности. Данные уровни определяются тегами: <h1>, <h2>,<h3>, <h4>, <h5>, <h6>.

Пример кода в Html с использованием заголовков всех уровней приведен на рисунке 4.1..

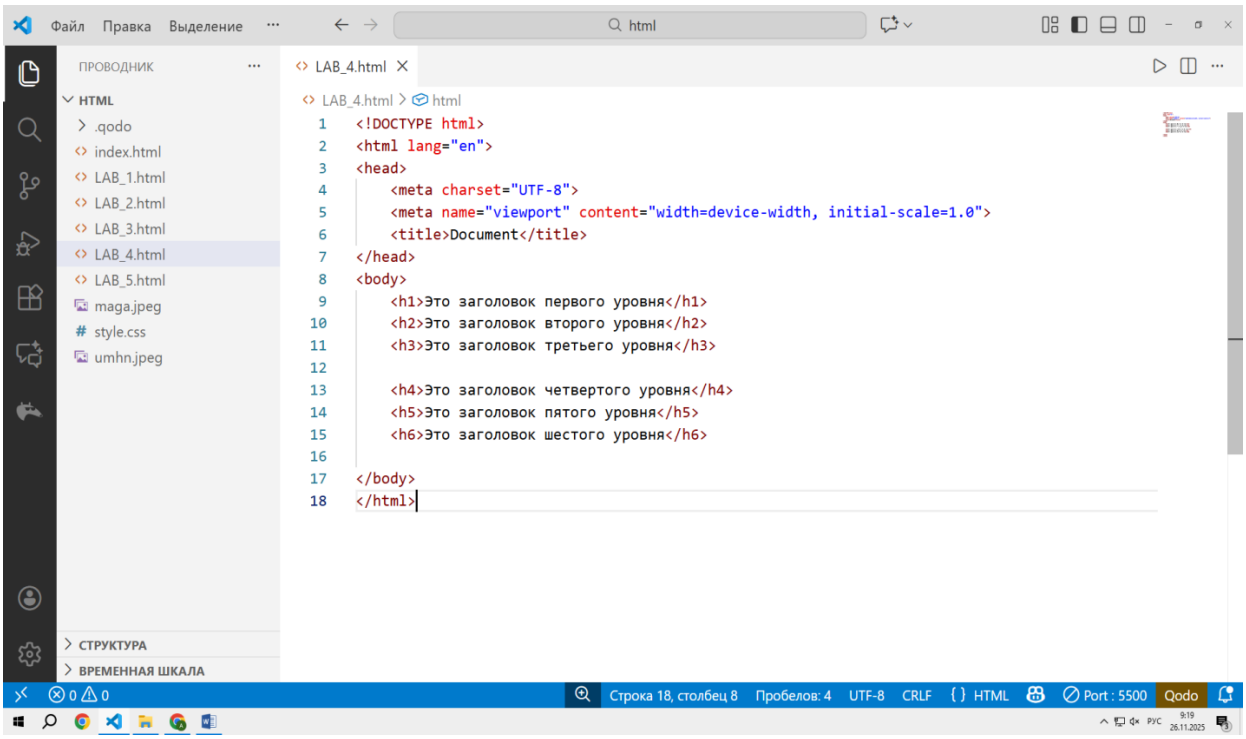



**Рисунок 4.1.1. Пример использования тегов <h></h> в Html-коде**

В окне браузера данная страница будет выглядеть как на рисунке 4.1.2.

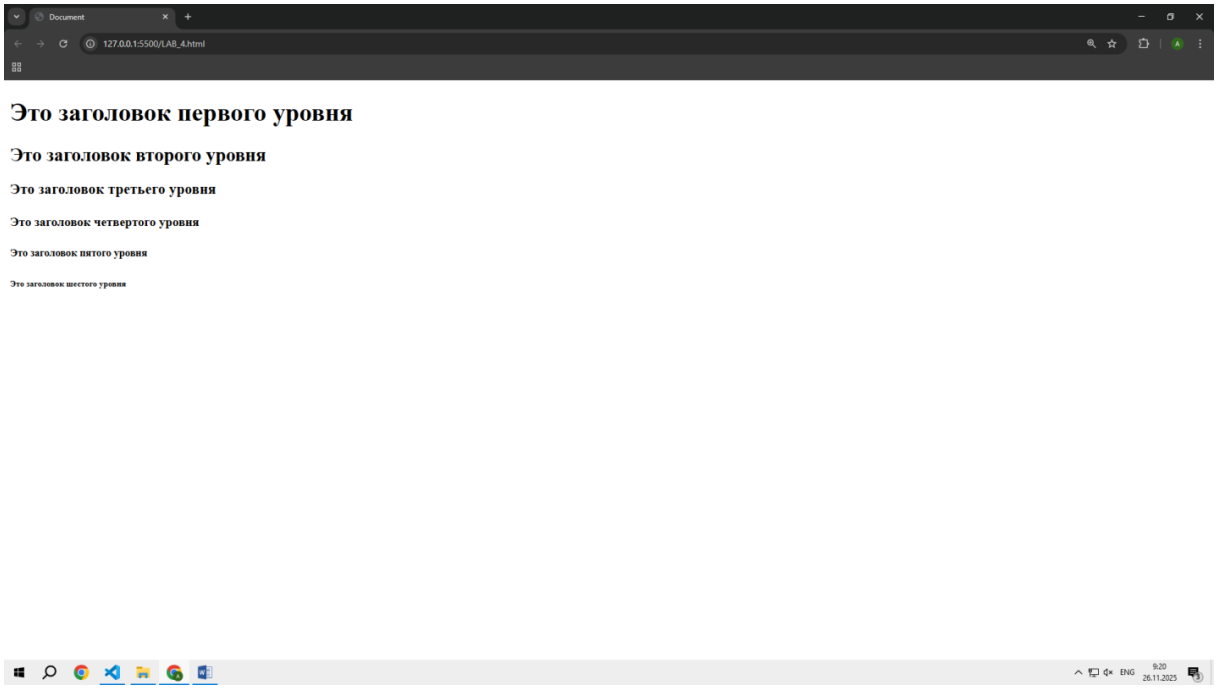



**Рисунок 4.1.2. Html-страница в окне браузера**

Чаще всего в практике используются только первые три уровня заголовков (h1, h2,h3). Здесь стоит также отметить, что не рекомендуется использовать тег заголовков в случае, когда вы просто хотите выделить какой-то фрагмент текста жирным, так как поисковые системы уделяют большое внимание заголовкам, т.к. используют их для индексации структуры и контента сайта.

## 4.2. Линии

Чтобы добавить на страницу горизонтальную линию используется тег <hr>. Этот визуальный компонент позволяет сделать на сайте разделение содержимого на блоки.

Структура Html-кода с использованием тега <hr> приведена на рисунке 4.2.1.

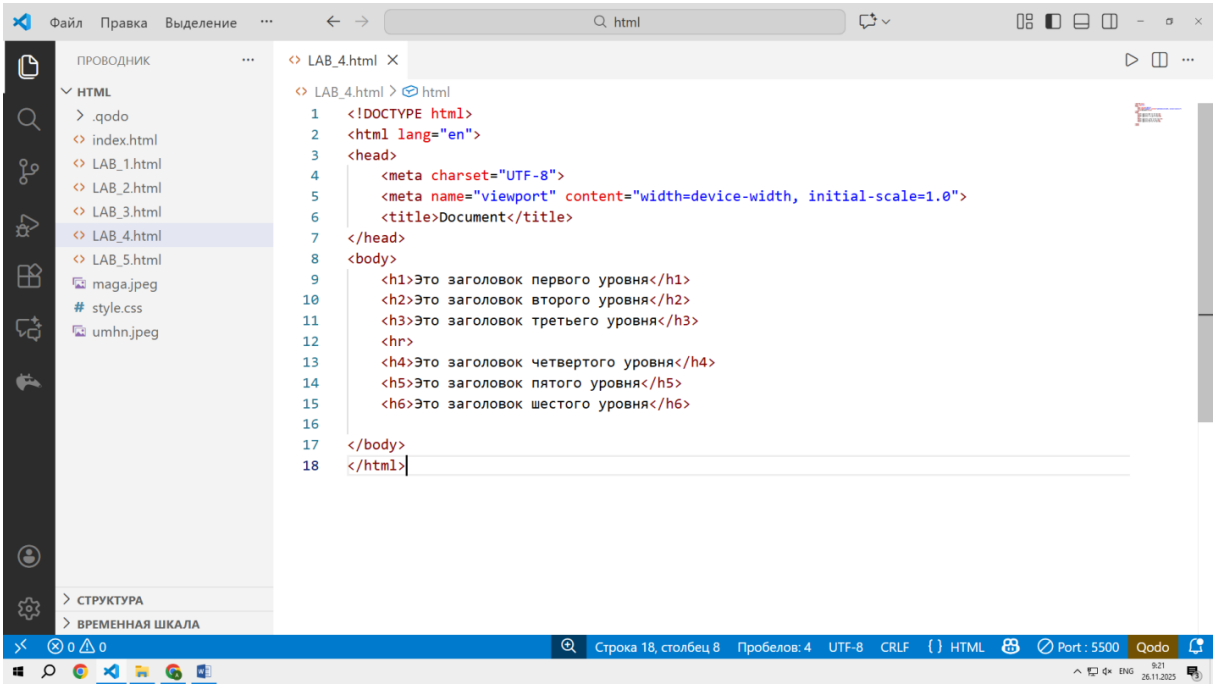



**Рисунок 4.2.1. Структура Html-кода с использованием тега <hr>**

Данный тег является одиночным, т.е. не требует конечного, закрывающего, тега.

В окне браузера эта страница будет выглядеть как на рисунке 4.2.2.

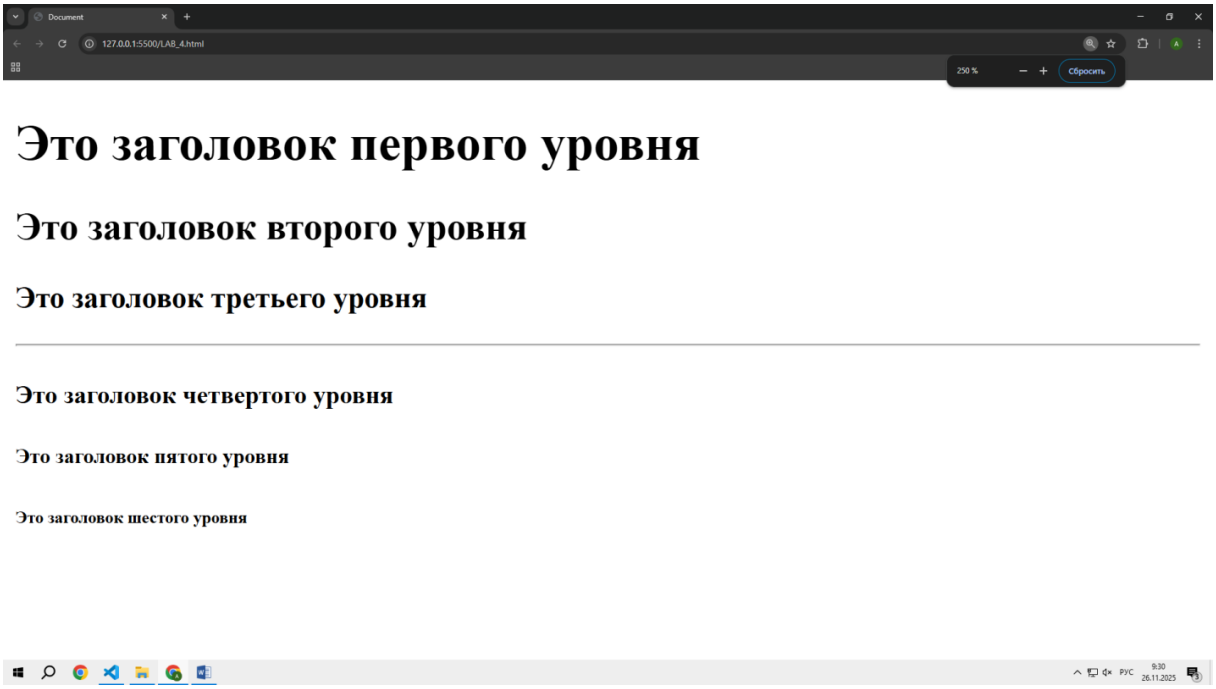



**Рисунок 4.2.2. Html-страница в окне браузера**

## 4.3. Комментарии

В составлении документа также помогают комментарии, они дают возможность добавлять описания, делать заметки, напоминания и т.д. Они могут использоваться также и в отладке кода, к примеру, если вы хотите временно убрать со страницы какой-то фрагмент, не удаляя его, можно данный фрагмент просто поместить в комментарии.

Сами комментарии в структуре Html-кода записываются, также как и теги, в угловые скобки (рисунок 4.3.1).

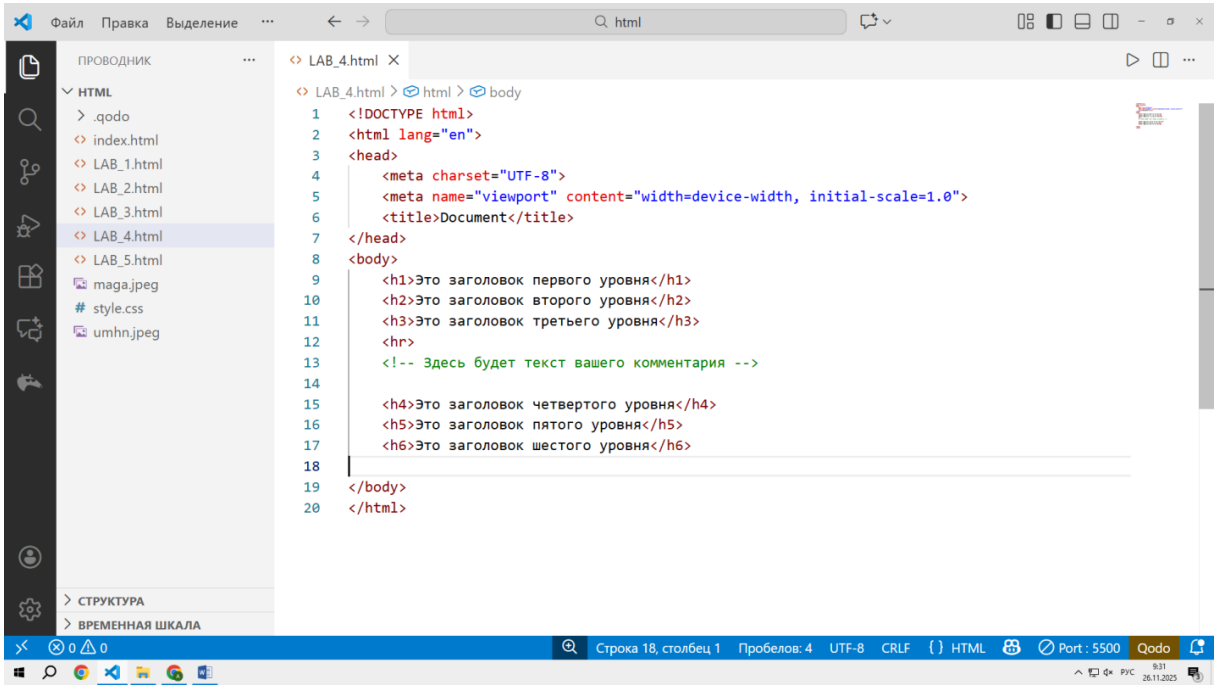



**Рисунок 4.3.1. Структура Html-кода с использованием комментариев**

Комментарии игнорируются браузером и поэтому никак не будут отображаться на самой странице.

## 4.4. Символы

Ранее уже не раз было отмечено, что теги в Html-коде записываются в угловых скобках. Напрашивается вопрос, как браузер узнает, когда теги, например <h1></h1>, являются управляющими, то есть делают текст заголовком, а когда их нужно напечатать просто как текст.

На самом деле всё, что находится в угловых скобках, браузер считает управляющими тегами всегда. В случае если нужно отобразить эти символы

на экране, необходимо записать их особым образом, используя специальные символы Html.

Приведем несколько основных специальных символов в таблице 4.4.1.

*Таблица 4.4.1*
*Специальные символы Html и их значения*

| Символ | Значение |
|---|---|
| <lt; | знак "меньше" |
| <gt; | знак "больше" |
| <nbsp; | непрерывный пробел |
| <copy; | знак "copyright" |
| <amp; | знак "амперсанд" |
| <laquo; | левая открывающаяся кавычка |
| <raquo; | правая закрывающаяся кавычка |
| <uarr; | стрелка вверх |
| <rarr; | стрелка вправо |
| <darr; | стрелка вниз |
| <larr; | стрелка влево |
| <harr; | двунаправленная стрелка влево-вправо |

Пример использования нескольких специальных символов из таблицы в коде Html приведен на рисунке 4.4.1.

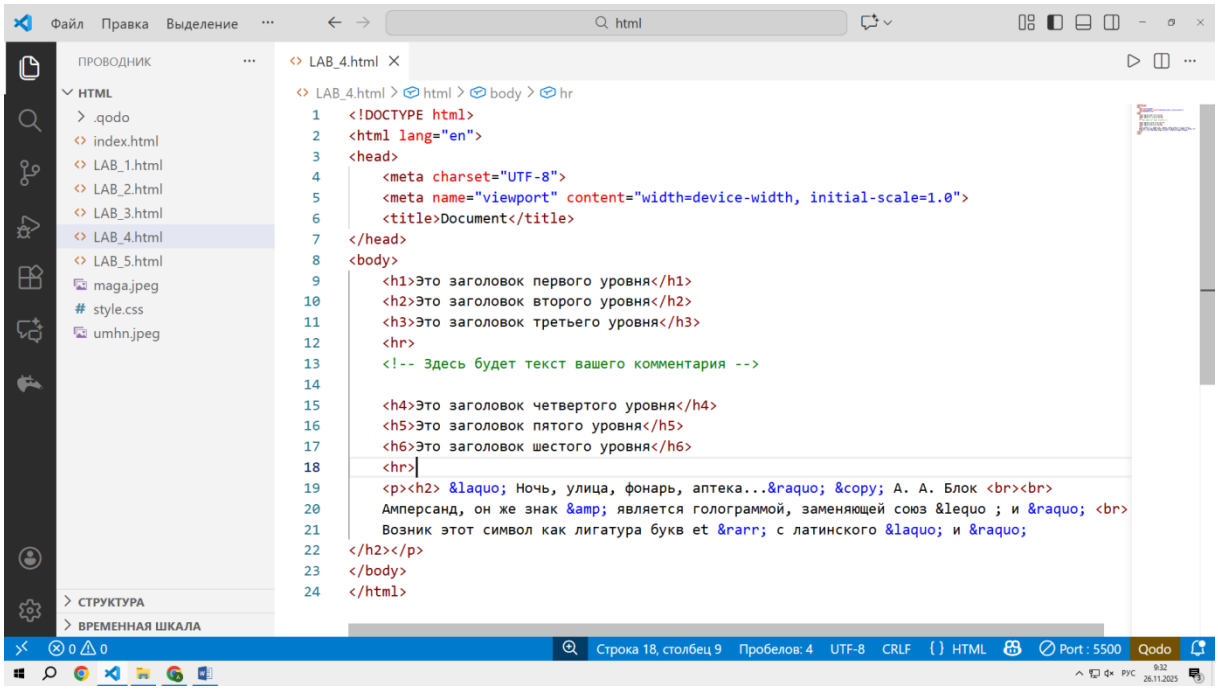



**Рисунок 4.4.1. Пример использования специальных символов в Html-коде**

В окне браузера эта страница будет выглядеть как на рисунке 4.4.2.

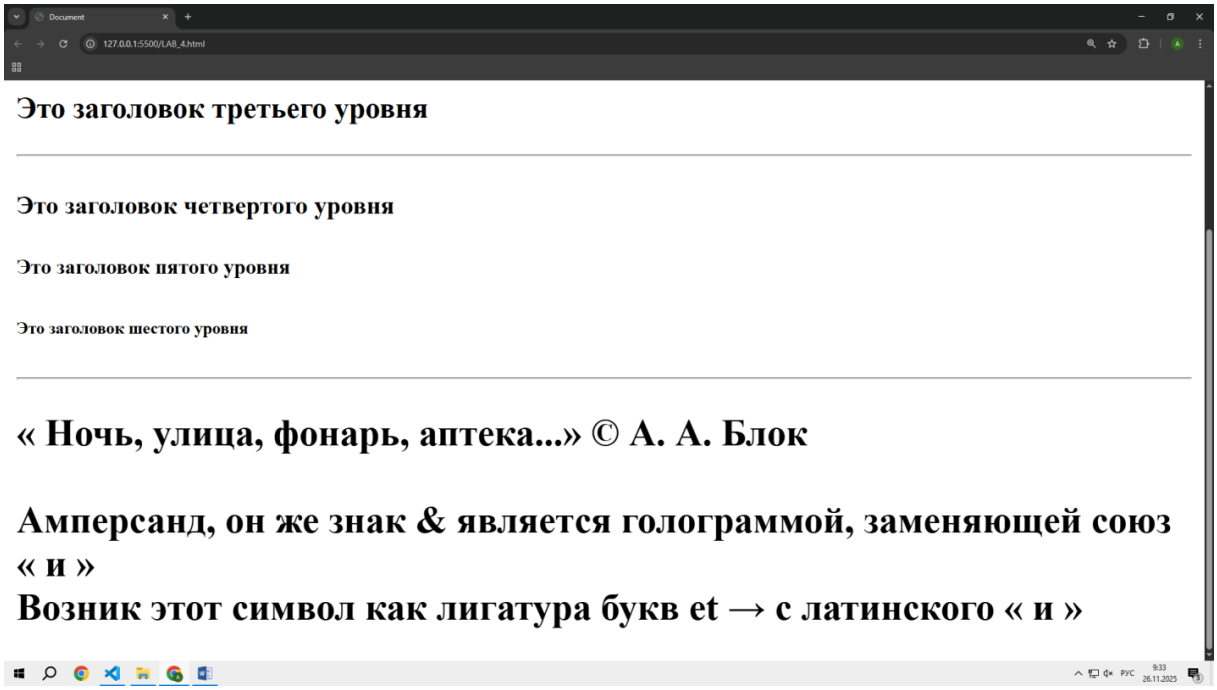



**Рисунок 4.4.2. Html-страница в окне браузера**

# Лабораторная работа №5

## Элементы и атрибуты в HTML

Html-элементы - это «строительные блоки» для любого Html-документа. Они представляют собой комбинацию начального и конечного тегов, обрамляющих содержимое. Браузер будет корректно отображать элемент, даже если вы не укажете конечный тег, но в этом случае, все содержимое страницы будет находиться внутри открытого тега.

Html - документ может состоять из вложенных друг в друга элементов. Можно сказать, что в языке Html сценарии пишутся с помощью элементов внутри самих элементов. Многие элементы имеют также и атрибуты.

Атрибуты содержат дополнительную информацию об элементе или теге, а также изменяют их. Большинство атрибутов имеют значения, которые меняют атрибут.

Пример использования атрибутов на примере тегов  в Html-коде приведен на рисунке 5.1.

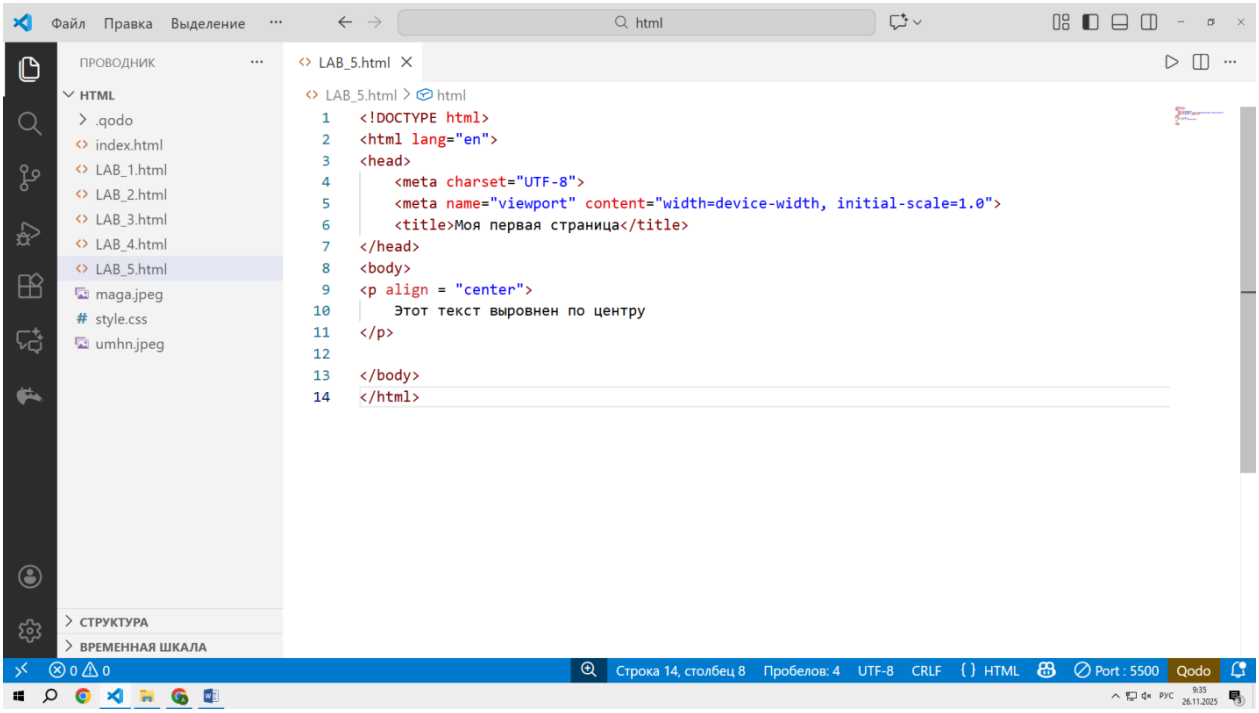



**Рисунок 5.1. Пример Html-кода с использованием атрибутов**

В окне браузера эта страница будет выглядеть как на рисунке 5.2.

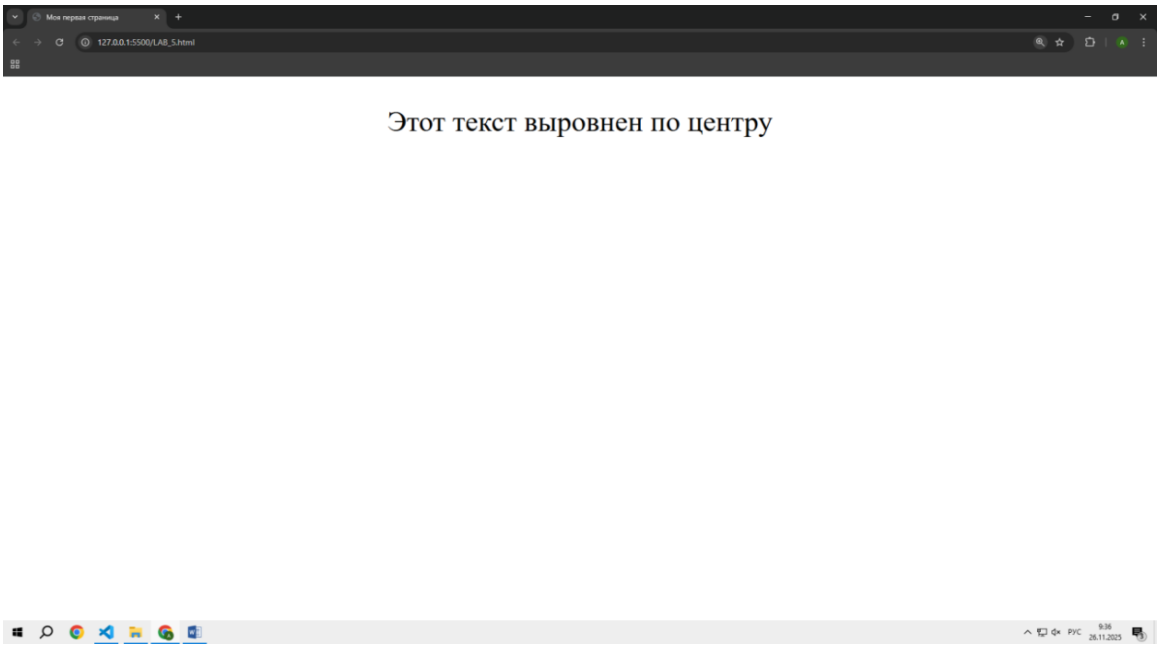



**Рисунок 5.2. Html-страница в окне браузера**

Атрибуты всегда пишутся в открывающем теге и имеют парную структуру (название атрибута → значение атрибута). В примере, приведенном на рисунке 5.1., **align** это название атрибута, а **center**-значение. Значение атрибута всегда записывается в кавычках.

Рассмотрим единицы измерения в атрибутах. В качестве примера добавим на нашу страницу горизонтальную линию шириной 50 пикселей (рисунок 5.3).

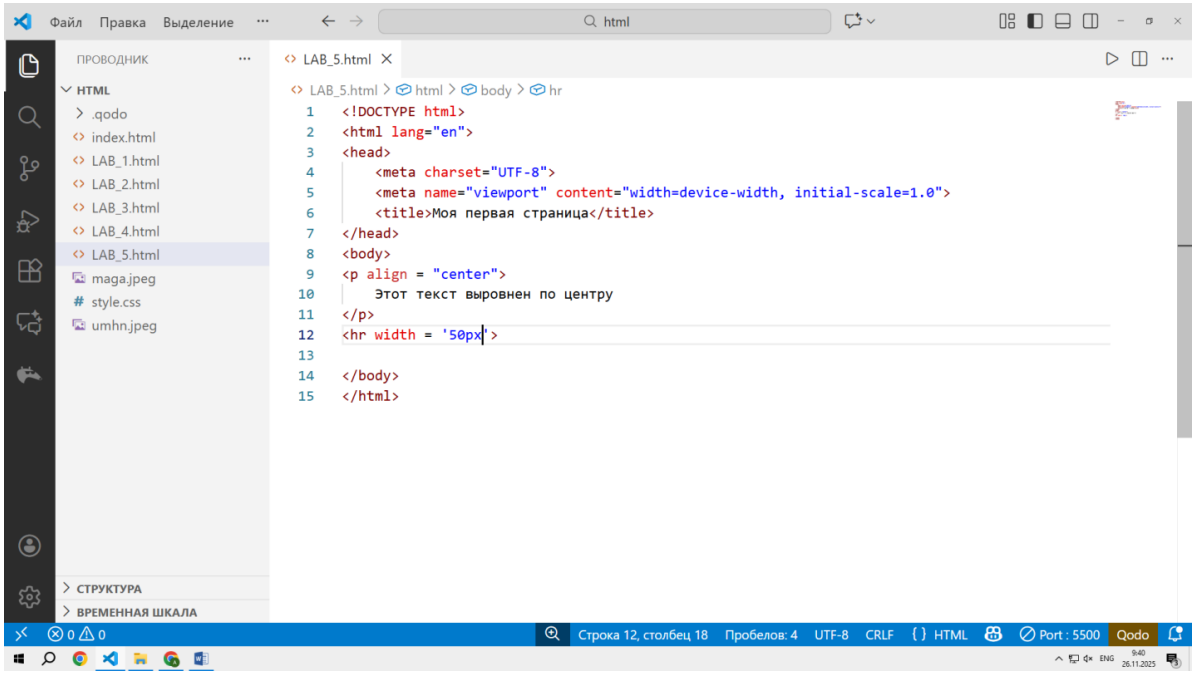



**Рисунок 5.3. Структура Html-кода с использованием атрибута width тега <hr>**

В окне браузера данная страница будет выглядеть как на рисунке 5.4.

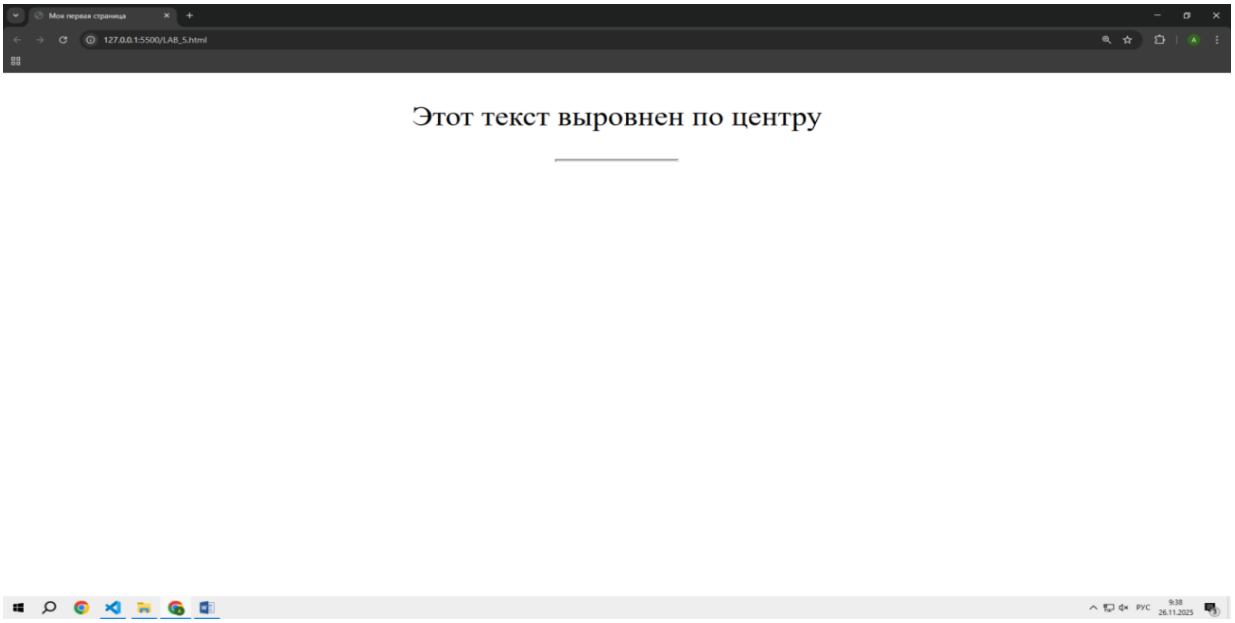



**Рисунок 5.4. Html-страница в окне браузера**

Ширина элемента может быть также определена в процентах (рисунок 5.5).

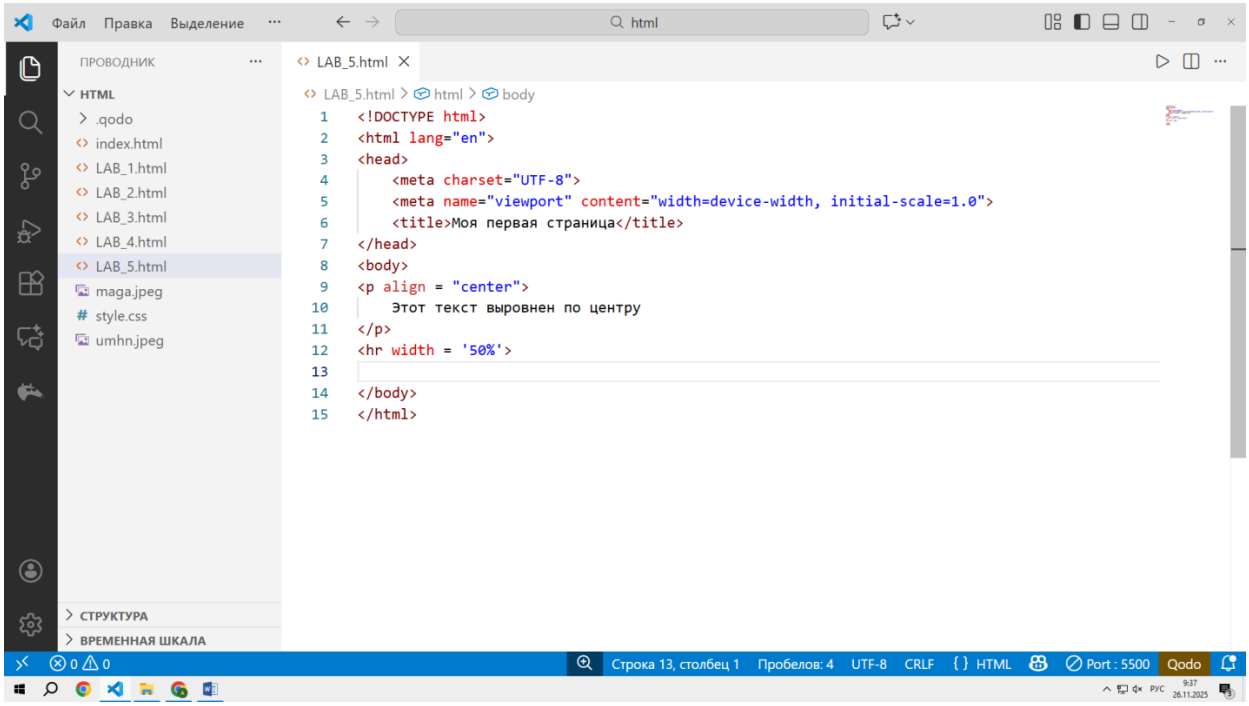



**Рисунок 5.5. Структура Html-кода с использованием атрибута width тега <hr> (значение атрибута задано в процентах)**

В окне браузера данная страница будет выглядеть как на рисунке 5.6.

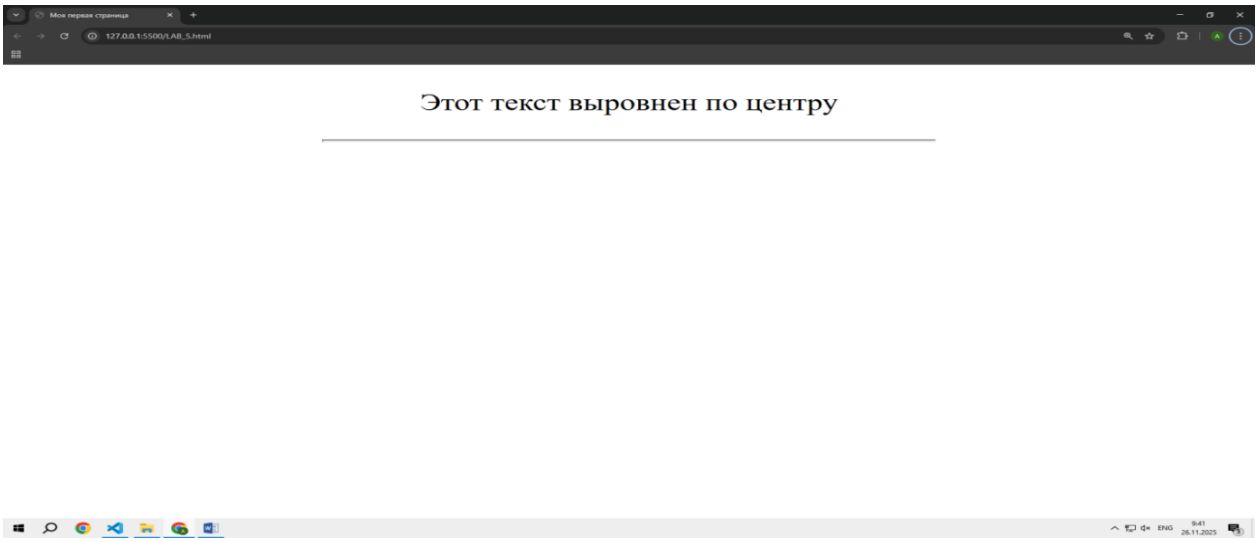



**Рисунок 5.6. Html-страница в окне браузера**

В данном случае ширина линии составляет 50% от количества пикселей на экране монитора.

Для закрепления материала, рассмотрим еще один пример. Необходимо разместить на странице текст стихотворения А.А. Блока «Ночь, улица, фонарь, аптека» в соответствии с параметрами, приведенными в таблице 5.1.

*Таблица 5.1*
*Параметры размещения текста стихотворения на странице*

| **Строки стихотворения** | **Параметры размещения на странице** |
|---|---|
| Ночь, улица, фонарь, аптека,<br>Бессмысленный и тусклый свет.<br>Живи еще хоть четверть века -<br>Все будет так. Исхода нет. | Выравнивание: по центру страницы<br>Начертание шрифта: курсивное<br>Цвет шрифта: красный<br>Размер шрифта: 14 пт<br>Дополнительные параметры:<br>сохранить авторскую разбивку на строки |
| Умрешь - начнешь опять сначала<br>И повторится всё, как встарь:<br>Ночь, ледяная рябь канала,<br>Аптека, улица, фонарь. | Выравнивание: по правому краю страницы<br>Начертание шрифта: жирное<br>Цвет шрифта: синий<br>Размер шрифта: 18пт<br>Дополнительные параметры:<br>сохранить авторскую разбивку на строки |
| Александр Александрович Блок | Выравнивание: по центру страницы<br>Цвет: значение параметра по умолчанию (черный)<br>Начертание шрифта: курсивное, жирное<br>Размер шрифта: 16 пт<br>Дополнительные параметры: перед ФИО автора поставить символ «копирайт» |

Соответствие атрибутов и их значений заданным параметрам приведено в таблице 5.2.

*Таблица 5.2*
*Соответствие атрибутов и их значений заданным параметрам*

| Параметры | Атрибуты и значения |
|---|---|
| Цвет шрифта | Тег <p>, атрибут - style, значение - color: red |
| Размер шрифта | Тег <font>, атрибут - size, значение - 3 (для шрифта 14 пт) и 5 (для шрифта 18пт) |
| Выравнивание на странице | Тег <p>, атрибут - align, значение - center (для выравнивания по центру) и right(для выравнивания по правому краю) |
| Начертание шрифта | Теги  - курсивное начертание,  - жирное начертание |
| Дополнительные параметры: сохранение авторской разбивки строк | Тег <br> - перенос строки |
| Дополнительные параметры: вставка символа «копирайт» | Специальный символ - © |

Структура Html-кода страницы с применением перечисленных в таблице 5.2 элементов и их атрибутов приведена на рисунке 5.7.

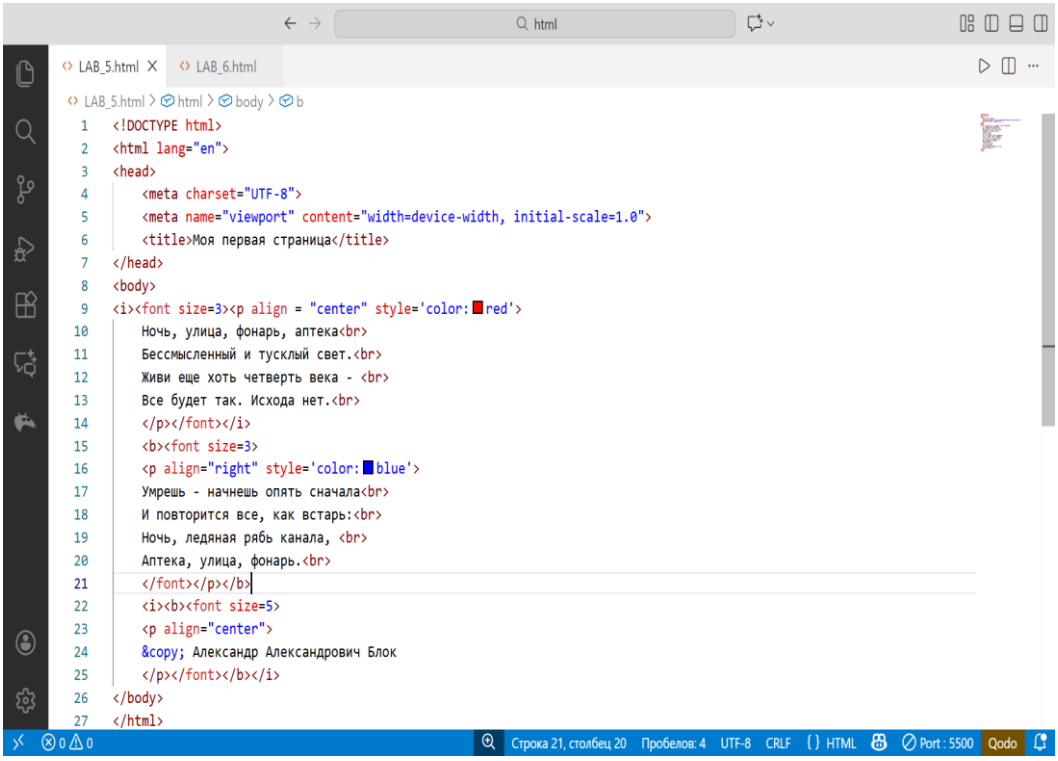



**Рисунок 5.7. Структура Html-кода с применением элементов и их атрибутов из таблицы 5.2**

В окне браузера данная страница будет выглядеть как на рисунке 5.7.

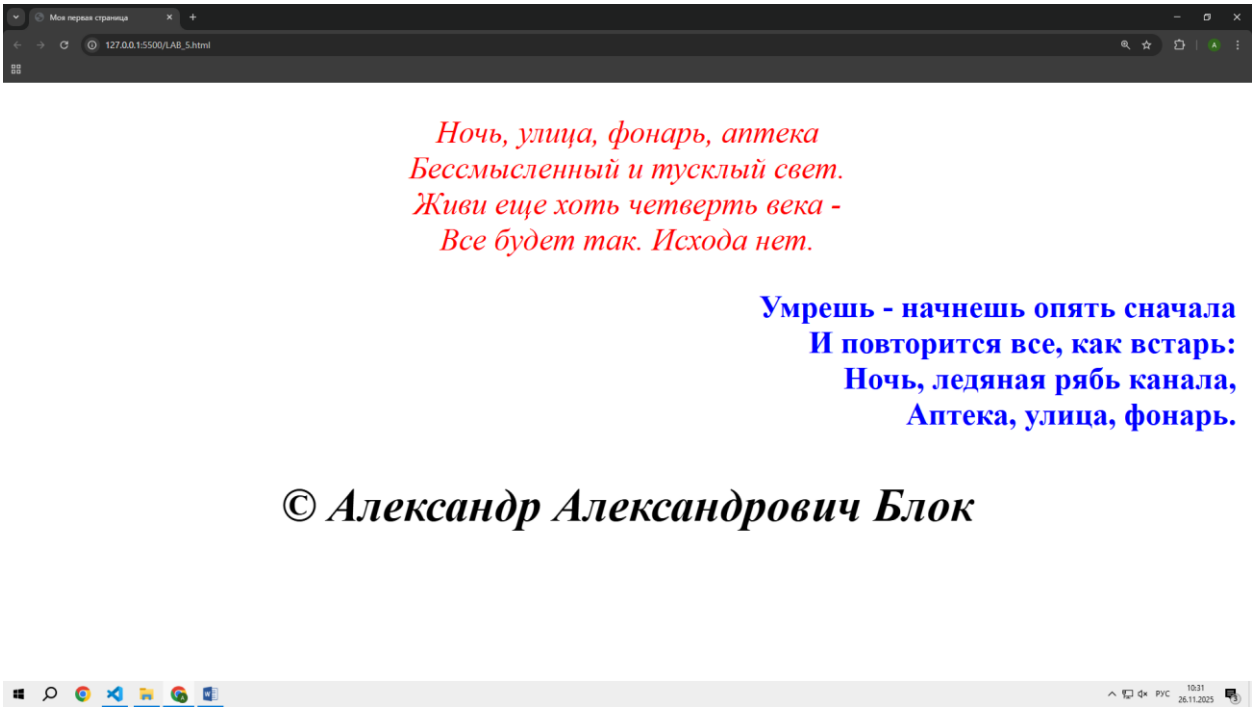



**Рисунок 5.7. Html-страница в окне браузера**

# Лабораторная работа №6

## Работа с изображениями в HTML

Комбинация изображений и текста на странице позволяет представлять пользователю информацию максимально эффективно.

Существует три формата графики, поддерживаемых всеми браузерами без специальных средств, это JPEG, PNGи GIF.

Для добавления в Html-документ изображения используется тег <img>. Данный тег используется только для вставки изображения, он содержит атрибуты и не требует закрывающего тега. Ссылка непосредственно на само изображение, или URL-адрес, добавляется с помощью атрибута src.

Если необходимо, то рисунок можно сделать ссылкой на другой файл, поместив тег <img> в контейнер <a>. При этом вокруг изображения отображается рамка, убрать которую можно, добавив атрибут borderco значением 0 в тег <img>.

Рисунки также могут применяться в качестве карт-изображений, когда картинка содержит активные области, выступающие в качестве ссылок. Такая карта по внешнему виду ничем не отличается от обычного изображения, но при этом оно может быть разбито на невидимые зоны разной формы, где каждая из областей служит ссылкой.

Синтаксис тега <img>выглядит следующим образом:

<imgsrc = “путь к изображению”>

Атрибуты, используемые в теге <img>, а также их описание приведены в таблице 6.1.

*Таблица 6.1*
*Атрибуты тега <img>*

| Название тега | Описание |
|---|---|
| Align | Определяет, как рисунок будет выравниваться по краю и способ обтекания текстом |
| Alt | Альтернативный текст для изображения |
| Border | Толщина рамки вокруг изображения |
| Height | Высота изображения |
| Hspace | Горизонтальный отступ от изображения до окружающего контента |
| Ismap | Говорит браузеру, что картинка является серверной картой-изображением |
| Longdesc | Указывает адрес документа, где содержится аннотация к картинке |
| Lowsrc | Адрес изображения низкого качества |
| Src | Путь к графическому файлу |
| Vspace | Вертикальный отступ от изображения до окружающего контента |
| Width | Ширина изображения |
| Usemap | Ссылка на тег<map>, содержащий координаты для клиентской карты-изображения |

Пример использования тега <img>с некоторыми атрибутами из таблицы приведен на рисунке 6.1.

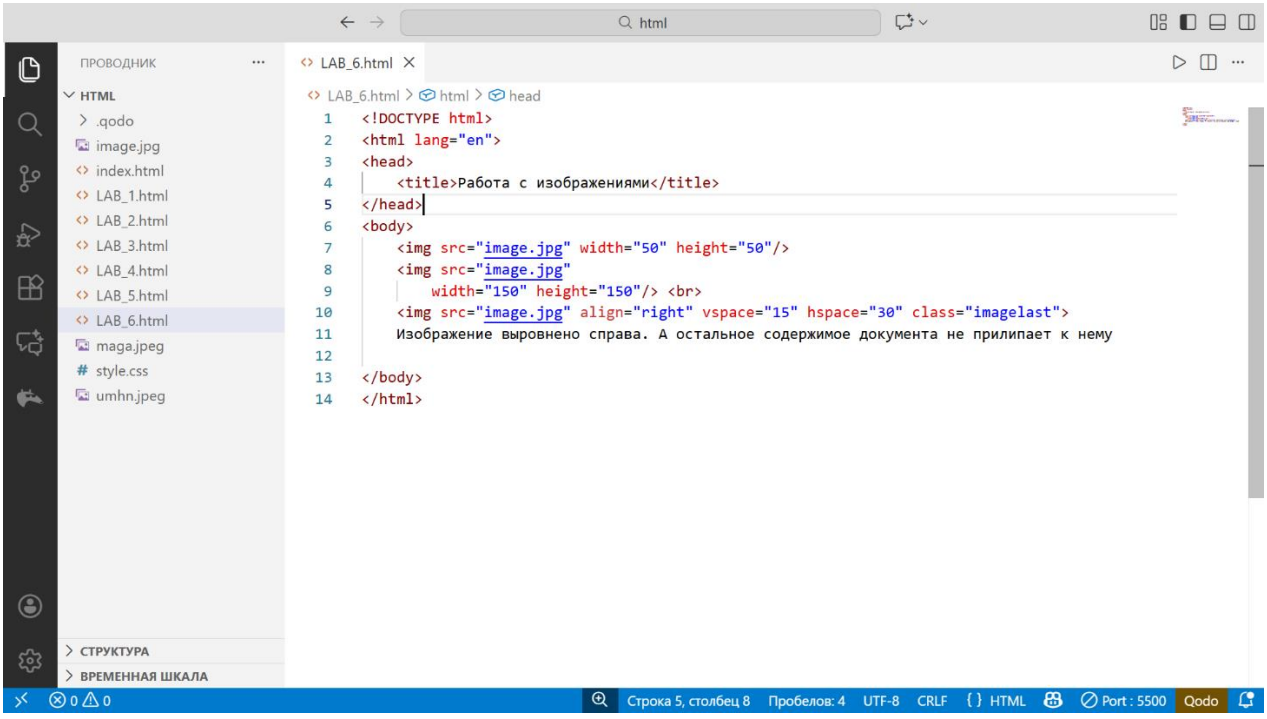



**Рисунок 6.1. Структура Html-кода с использованием тега <img> и его атрибутами**

В окне браузера данная страница будет выглядеть, как на рисунке 6.2.

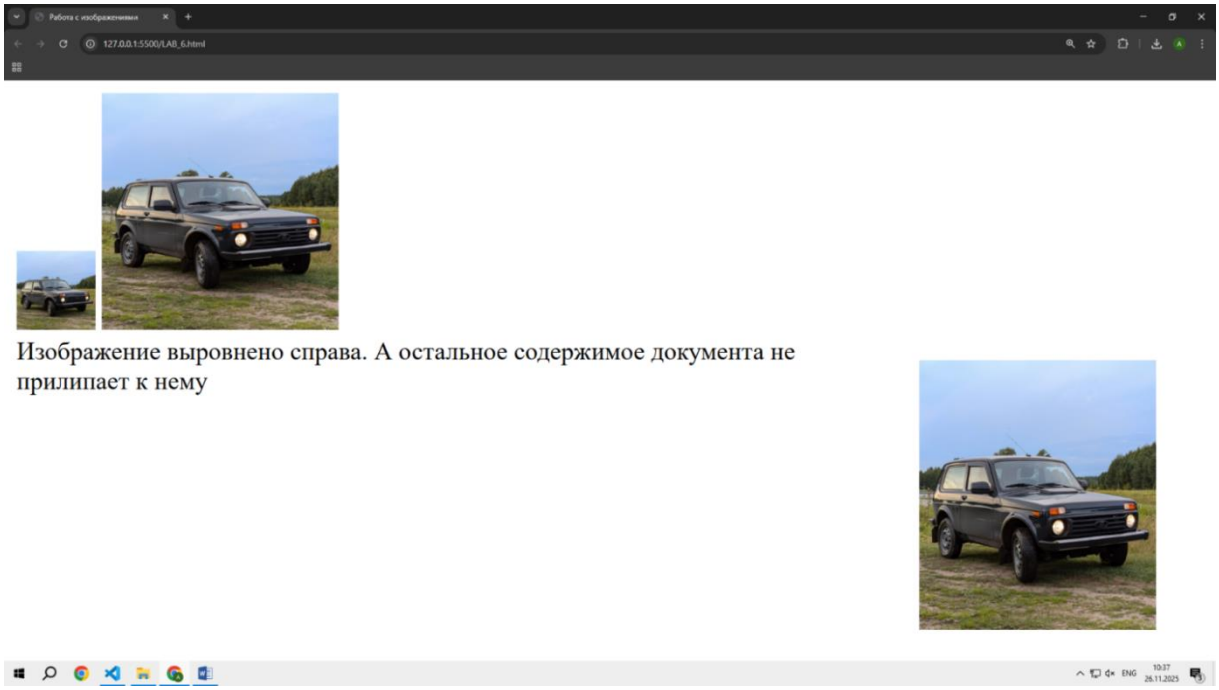



**Рисунок 6.2. Html-страница в окне браузера**

Рассмотрим возможность использования альтернативного текста. Он позволяет получить текстовую информацию о рисунке, если в браузере отключена возможность показа картинок или во время их загрузки. Такой текст появляется раньше самого изображения и дает представление об его содержании. Затем зарезервированное пустое поле заменяется картинкой. Для использования альтернативного текста служит атрибут alt. В случае, если картинка в браузере отображается, этот атрибут дает подсказку при наведении мыши, но не во всех браузерах. Структура Html-кода с использованием атрибута alt приведена на рисунке 6.3.

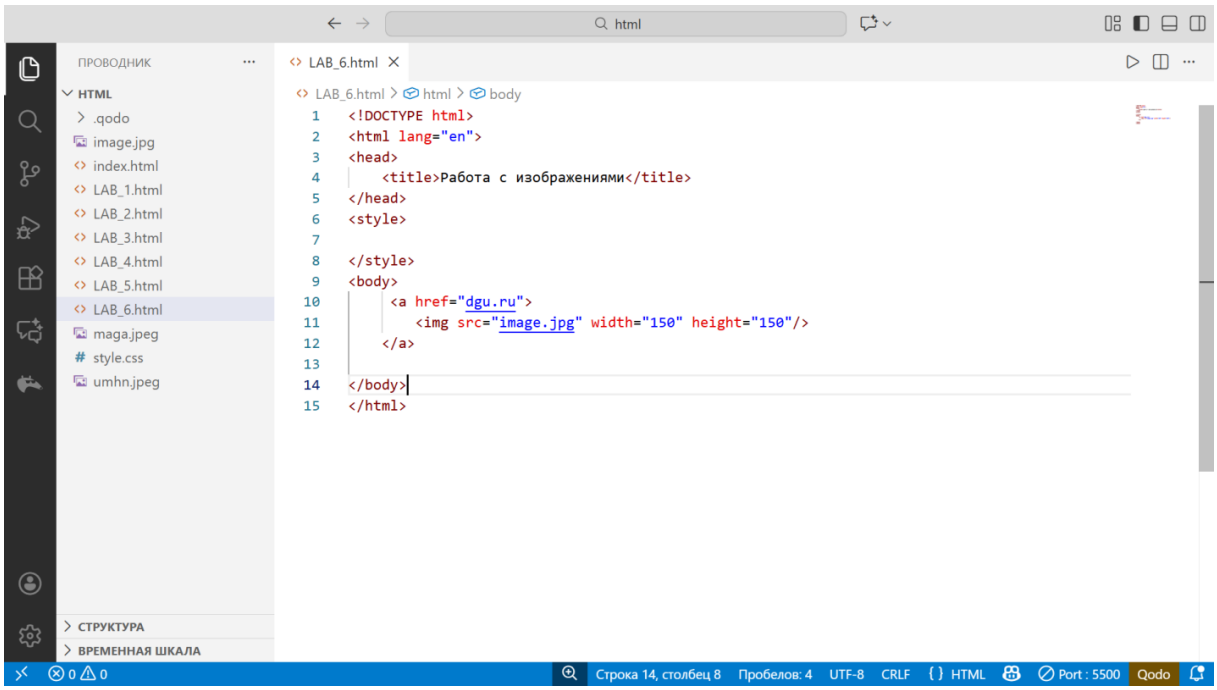



**Рисунок 6.3. Структура Html-кода с использованием атрибута alt тега <img>**

Чтобы использовать изображения в качестве ссылки необходимо поместить его внутрь тега-контейнера <a> (таблица 6.1). Для примера наше изображение будет ссылаться на сайт ДГУ.

Структура Html-страницы с использованием тега-контейнера <a>приведена на рисунке 6.3.

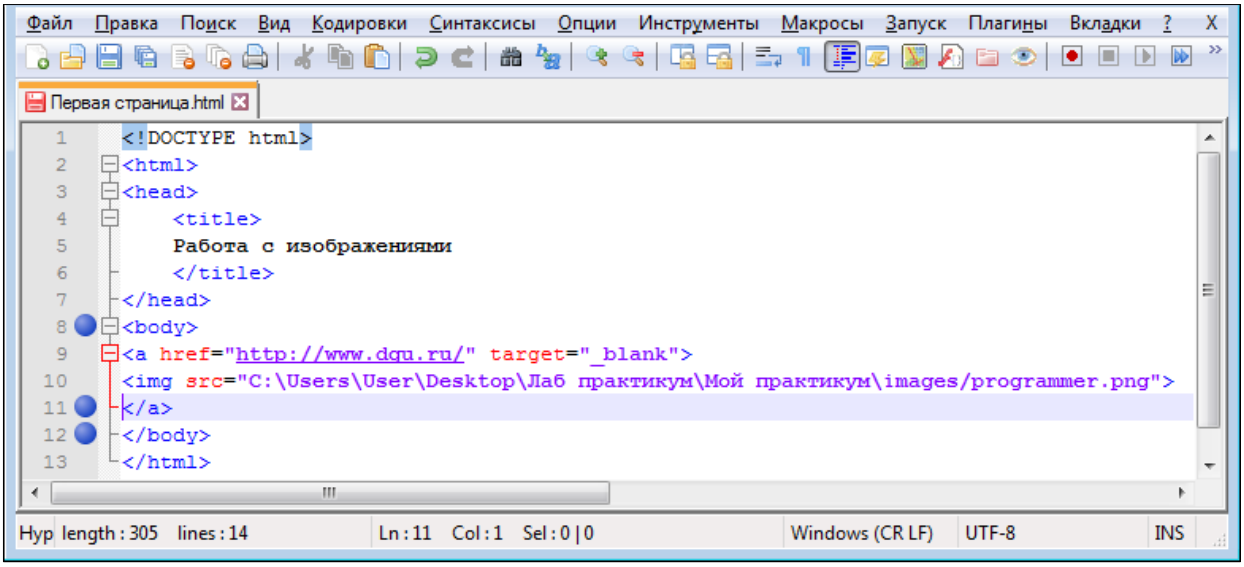



**Рисунок 6.3. Структура Html-страницы с использованием тега-контейнера <a>**

В окне браузера такая картинка ничем не будет отличаться от обычной, лишь при нажатии на нее откроется новое окно со страницей, на которую она ссылается.

Как видно из рисунка 6.3 в структуре кода, для того, чтобы сделать картинку ссылкой, мы использовали атрибуты href и target. Значением атрибута href, как правило, служит ссылка на нужный нам объект. А атрибут targetотвечает, как именно будет открываться объект, на который мы сослались, и имеет несколько значений. В данном примере было рассмотрено значение «_blank», что позволяет открыть страницу, которую мы прописали в href, в новом окне браузера.

Помимо обычной вставки изображения на страницах могут использоваться в качестве фона. Фоновое изображение заполняет собой все пространство элемента, для которого оно задано. Так, указав в теге <body>атрибут background со значением “fon.gif”, рисунком будет залито все окно браузера.

Структура Html-кода с использованием атрибута backgroundприведена на рисунке 6.4.

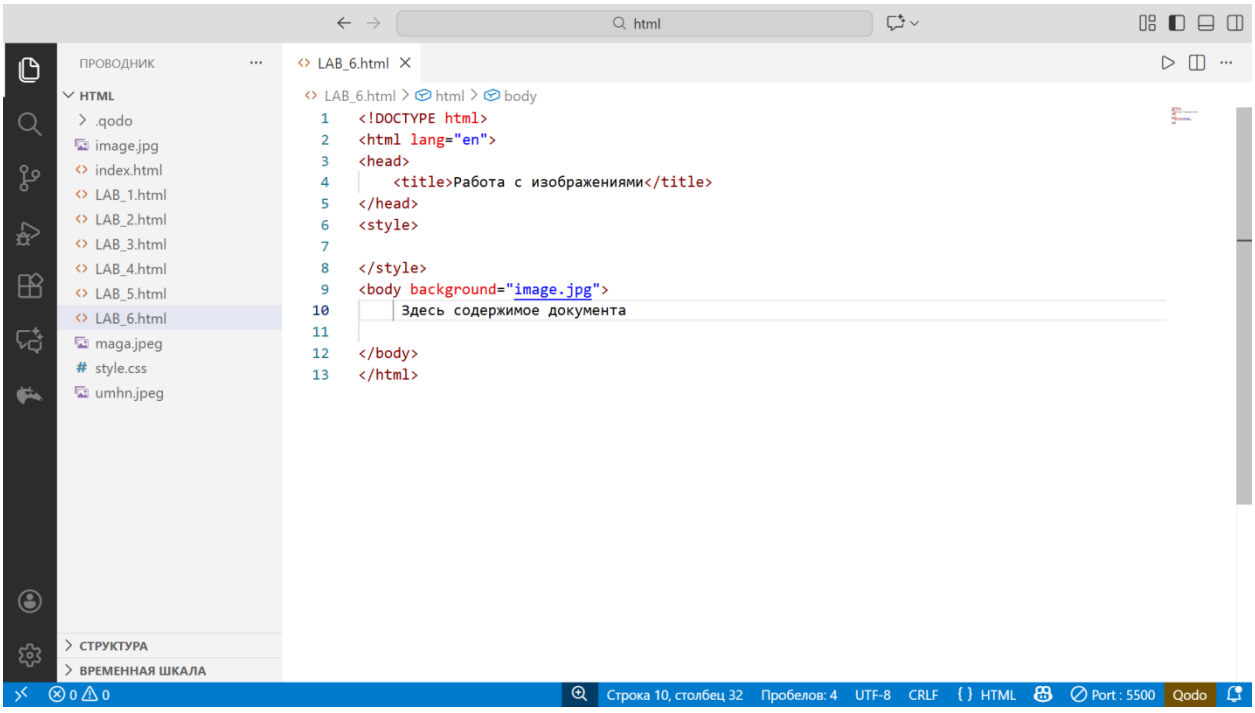



**Рисунок 6.4. Структура Html-кода с использованием атрибута background**

В окне браузера эта страница будет выглядеть как на рисунке 6.5.

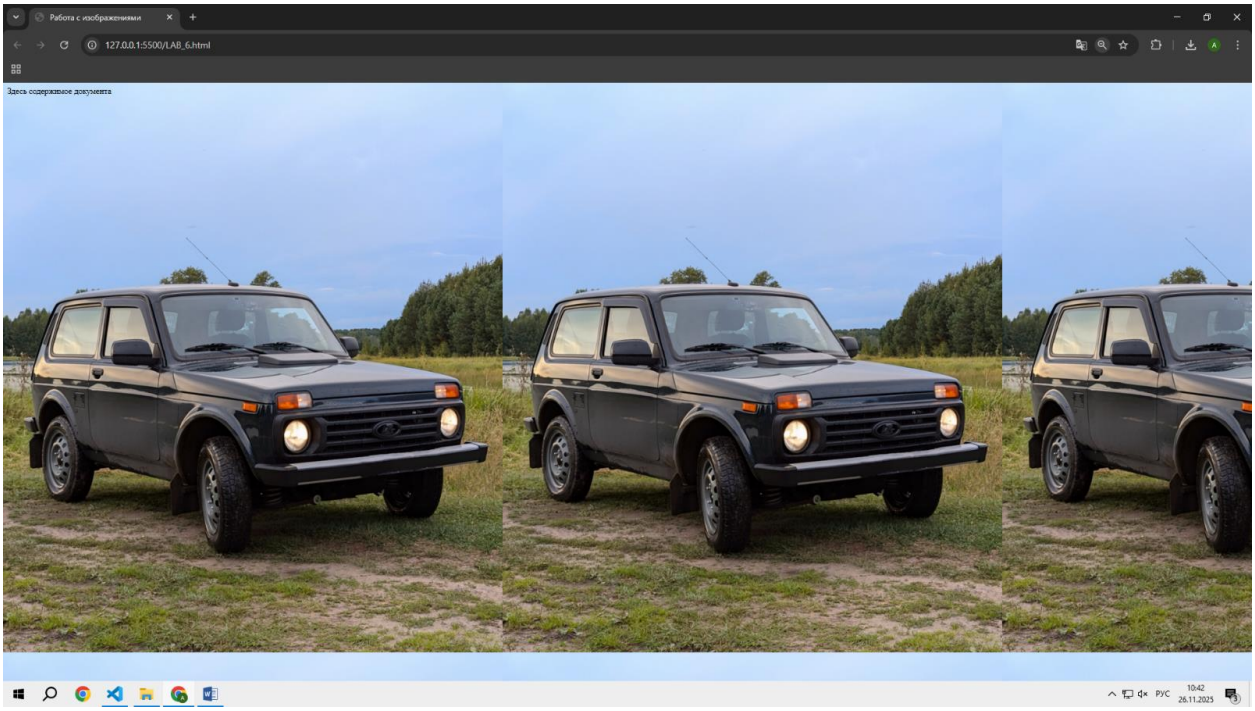



**Рисунок 6.5. Html-страница в окне браузера**

# Лабораторная работа №7

## Списки в HTML.

В Html выделяют три основных вида списков, это нумерованный список, маркированный список и список определений.

Нумерованные списки создаются с помощью тегов <ol></ol>, при этом каждый пункт этого списка задается тегами <li></li>. В данном случае номера элементов браузером вставляются автоматически и, при удалении какого-либо элемента из списка, остальные автоматически будут пересчитаны. Сама нумерация может быть произведена любым образом, это могут быть арабские цифры, цифры римского алфавита или буквы. По умолчанию нумерация ведется арабскими цифрами, что можно изменить с помощью параметра type, которому можно задать следующие значения:

1. type = “A” - прописные латинские буквы (A, B, C…);
2. type = “a” - строчные латинские буквы (a, b, c…);
3. type = “I” - большие римские цифры (I, II, III…);
4. type = “I” - маленькие латинские цифры (i, ii, iii…);
5. type = “1” - арабские цифры (1, 2, 3…).

Также у тега <ol>есть атрибут start, который позволяет задать начальное значение списка. Если нам нужно начать нумерацию не с единицы, то следует в качестве значения этого атрибута задать число, с которого начнется список. В случае если нумерация задается с помощью букв, то это число будет указывать на позицию буквы в алфавите.

У тега <li>есть атрибут value, который позволяет изменить номер данного элемента из списка. При этом нумерация всех последующих элементов из списка тоже меняется.

Структура Html-кода с использованием тегов для задания нумерованного списка приведена на рисунке 7.1.

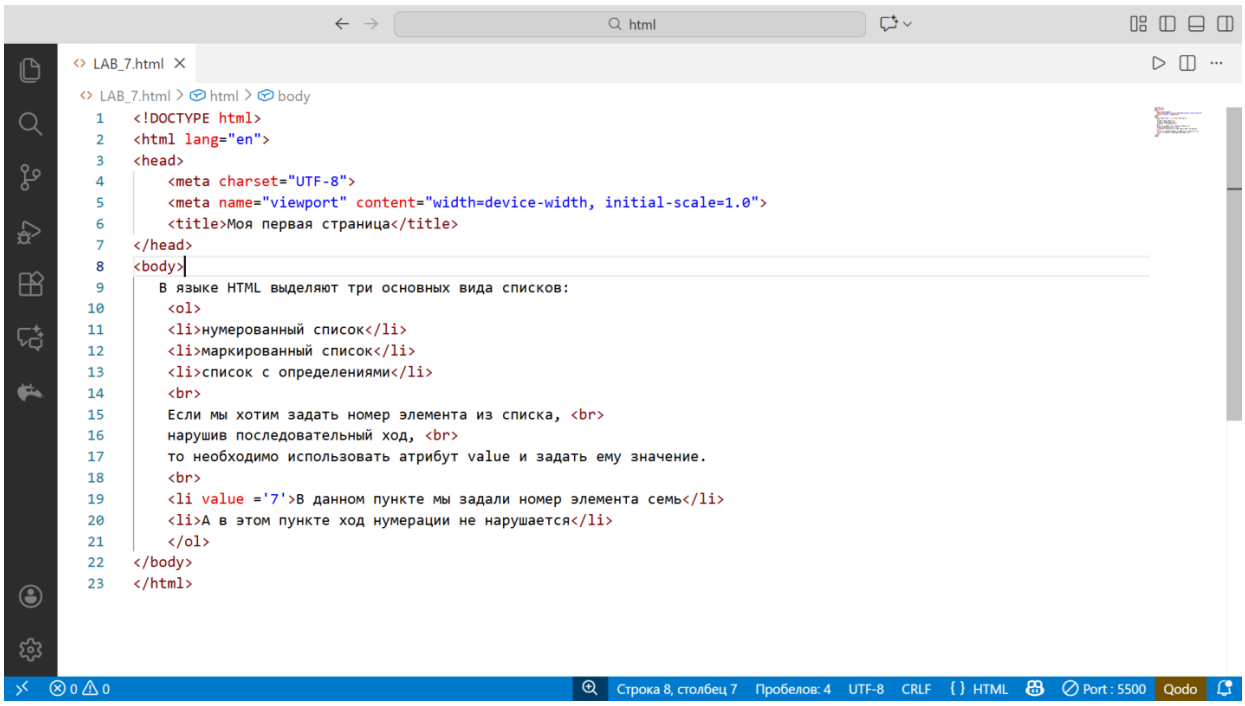



**Рисунок 7.1. Структура Html-кода с использованием тегов <ol>и <li>**

В окне браузера данная страница будет выглядеть как на рисунке 7.2.

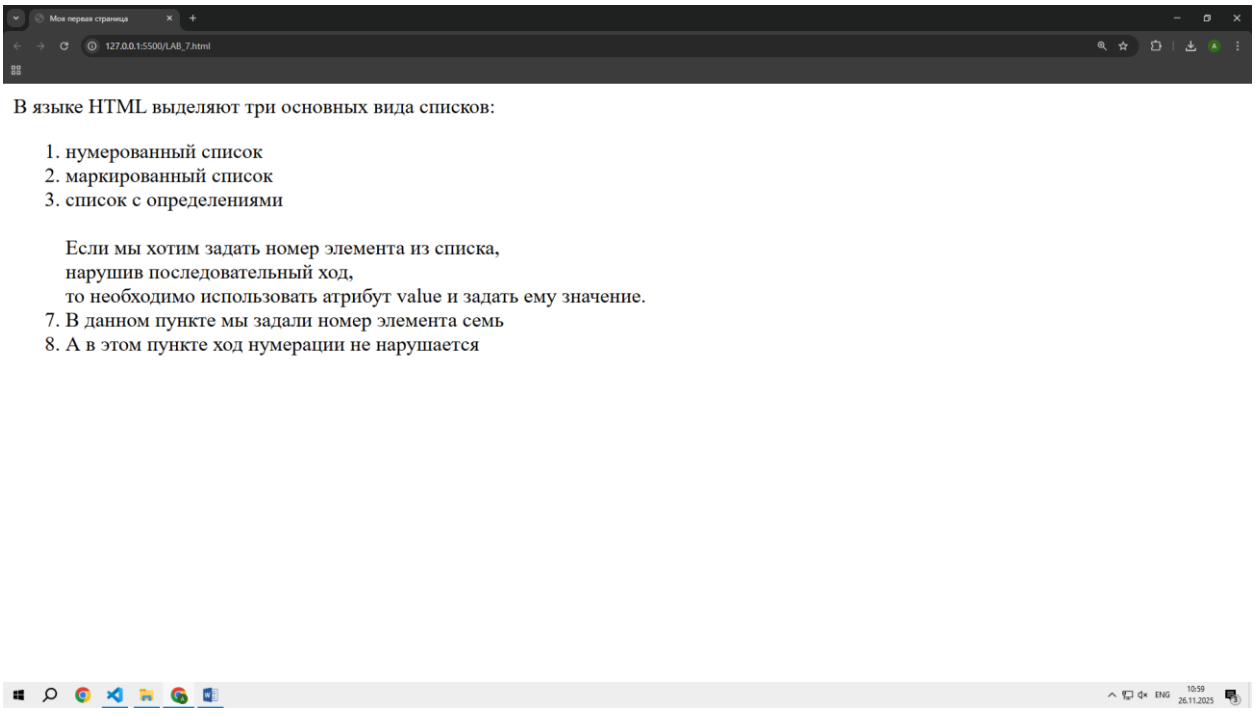



**Рисунок 7.2. Html-страница в окне браузера**

Маркированный список определяется тем, что перед каждым элементом списка вместо букв и цифр добавляется небольшой маркер. Маркеры могут принимать один из трех видов: круг (по умолчанию), окружность и квадрат.

Вид маркеров может незначительно различаться в разных браузерах, а также при смене шрифта и размера текста. Чтобы указать браузеру, что список может быть маркированным, используется парный тег-контейнер <ul></ul>. У данного тега есть атрибут type. Значения данного атрибута позволяют определять внешний вид маркеров:

1. type = “disk” - список с маркерами в виде закрашенного круга;
2. type = “circle” - список с маркерами в виде окружности;
3. type = “square” - список в маркерами в виде закрашенного квадрата.

Каждый пункт маркированного списка задается тегами <li></li>.

Структура Html-кода с использованием тегов для задания маркированных списков приведена на рисунке 7.3.

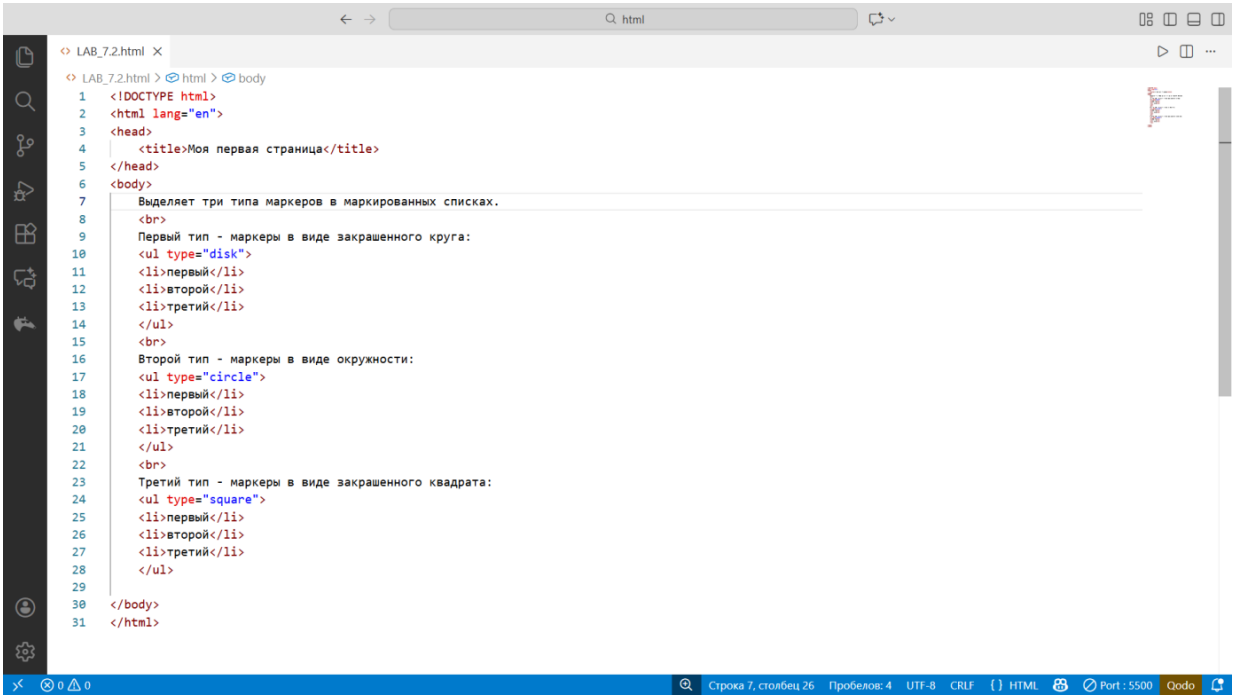



**Рисунок 7.3. Структура Html-кода с использованием тегов <ul>и <li>**

В окне браузера данная страница будет выглядеть как на рисунке 7.4.

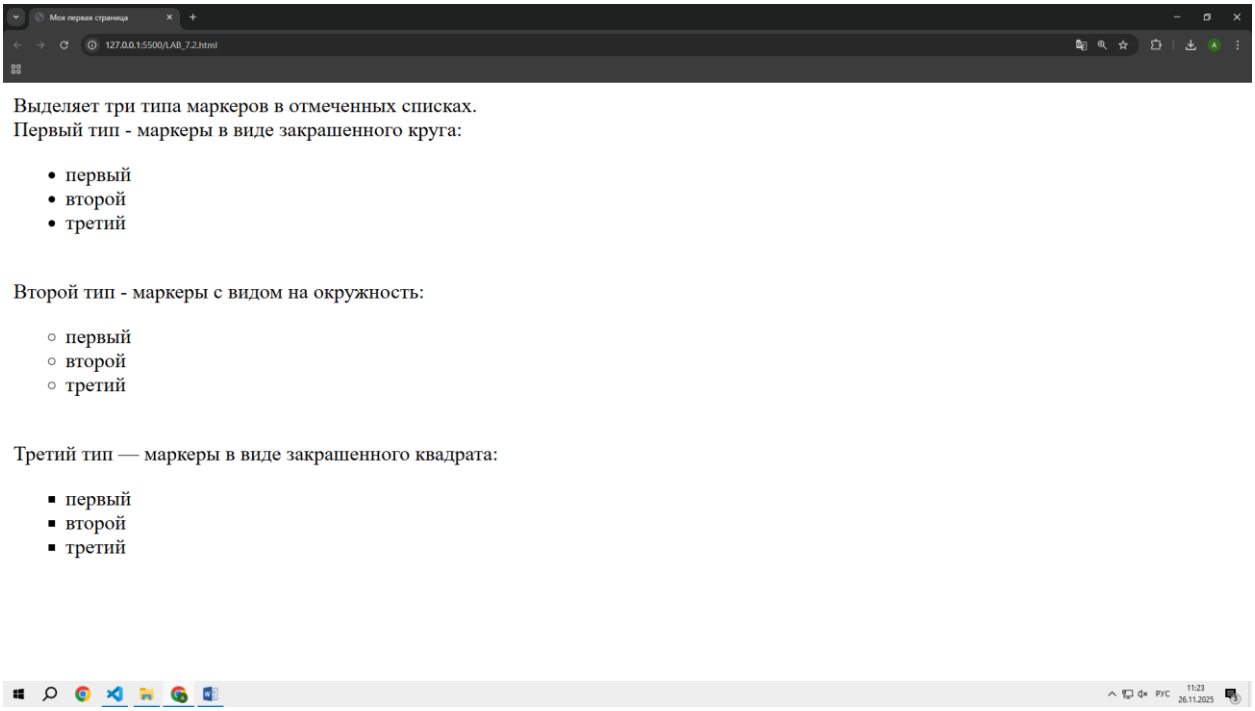



**Рисунок 7.4. Html-страница в окне браузера**

Как видно из рисунка 7.4 при создании маркированных списков браузер автоматически добавляет отступы сверху, снизу и слева от списка.

Третьим из основных видов списков в Html являются списки определений.

Списки определений используются в словарях и представляют собой специальным образом расположенные термины и их описания. Чтобы указать браузеру, что необходимо построить список определений, используются теги <dl></dl>. Каждый термин такого списка заключается в теги <dt></dt>, а его определение в теги <dd></dd>. У данных тегов есть следующие атрибуты:

1. class - стилевой класс;
2. id -имя;
3. lang - код языка;
4. style - определение встроенного стиля.

Структура Html-документа с использованием тегов для задания списков определений приведена на рисунке 7.5.

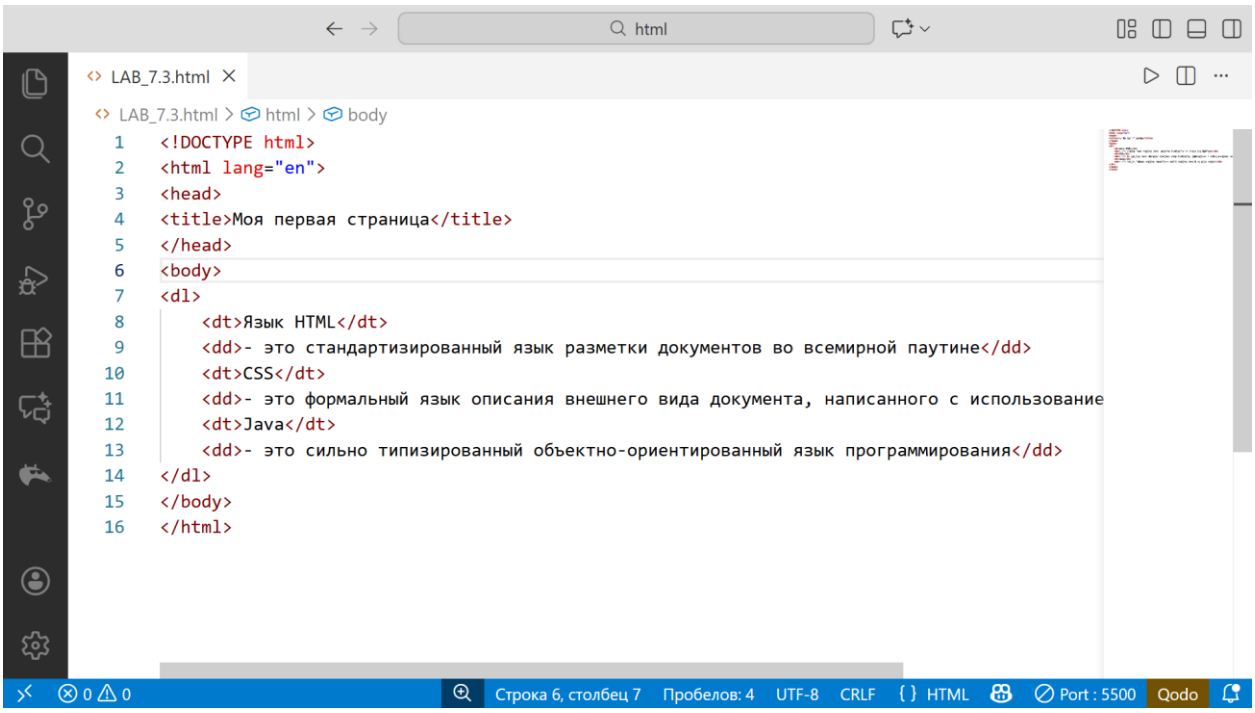



**Рисунок 7.5. Структура Html-кода с использованием тегов <dl>, <dt>, <dd>**

В окне браузера данная страница будет выглядеть как на рисунке 7.6.

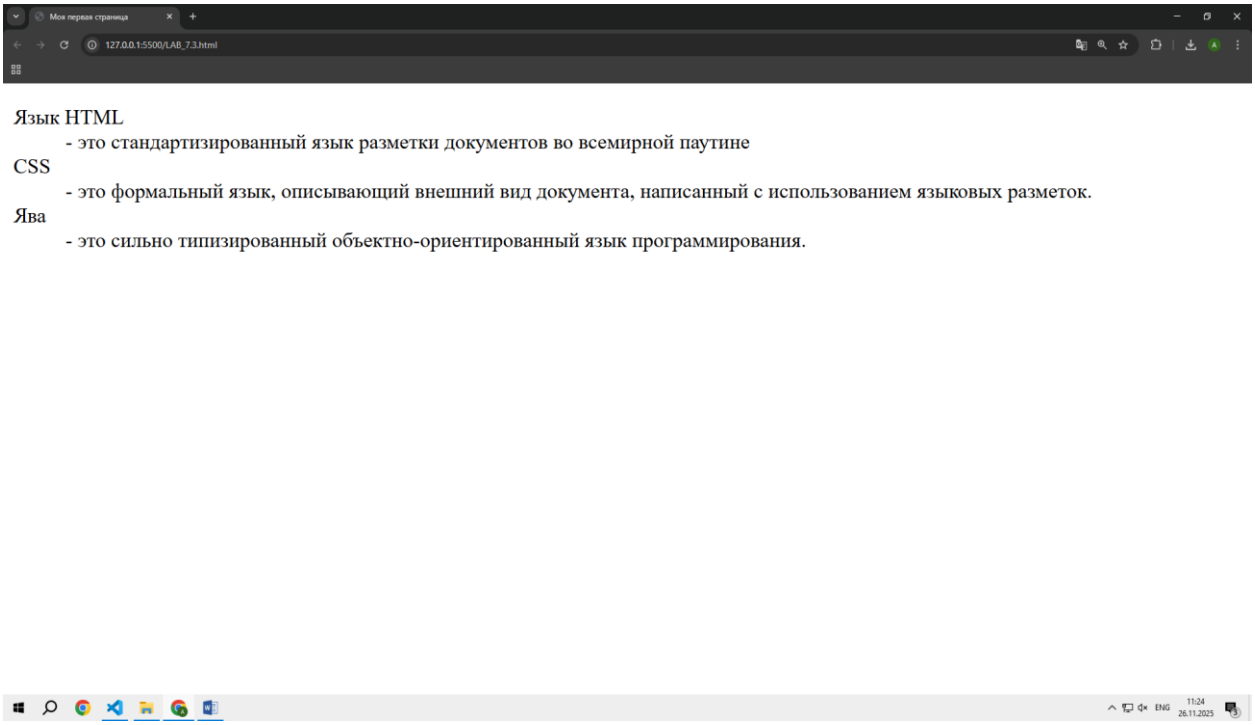



**Рисунок 7.6. Html-страница в окне браузера**

Для закрепления материала из данной лабораторной работы выполните следующее самостоятельное задание:

Используя вложение вышерассмотренных тегов, создайте список приведенный на рисунке 7.7.

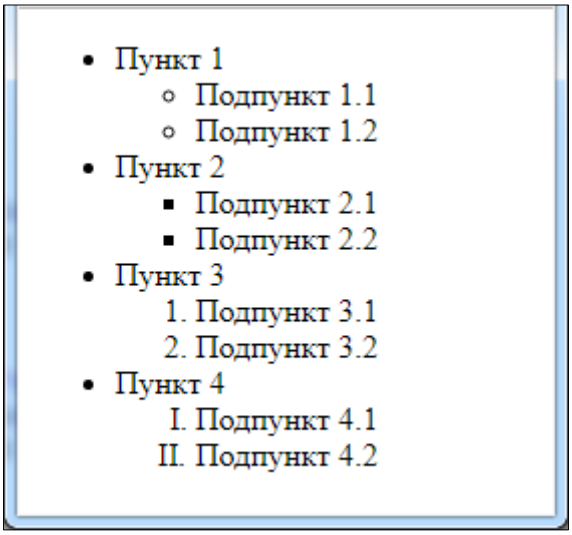



**Рисунок 7.7. Задание на самостоятельное выполнение**

# Лабораторная работа №8

## Работа с ссылками в HTML

Основой любого гипертекстового документа являются ссылки, позволяющие переходить с одной страницы на другую. Ссылки могут открывать файлы любого типа, независимо от того расположены они на этом же сайте или на другом.

Для того чтобы поместить ссылку на странице используется тег-контейнер <a>. Чтобы дать пользователю понять, что перед ним ссылка, необходимо выделить ее среди другого текста каким-либо образом. Содержимое тега <a> по умолчанию выделяется синим цветом и подчеркивается. У тега <a>есть ряд атрибутов.

Атрибут href используется для определения адреса документа, на который необходимо перейти. Принцип использования данного атрибута, его синтаксис, мы подробно рассмотрели в рамках лабораторной работы 6 при работе с изображениями. Тем не менее, для закрепления материала, приведем еще один пример структуры Html-кода с использованием атрибута href (рисунок 8.1).

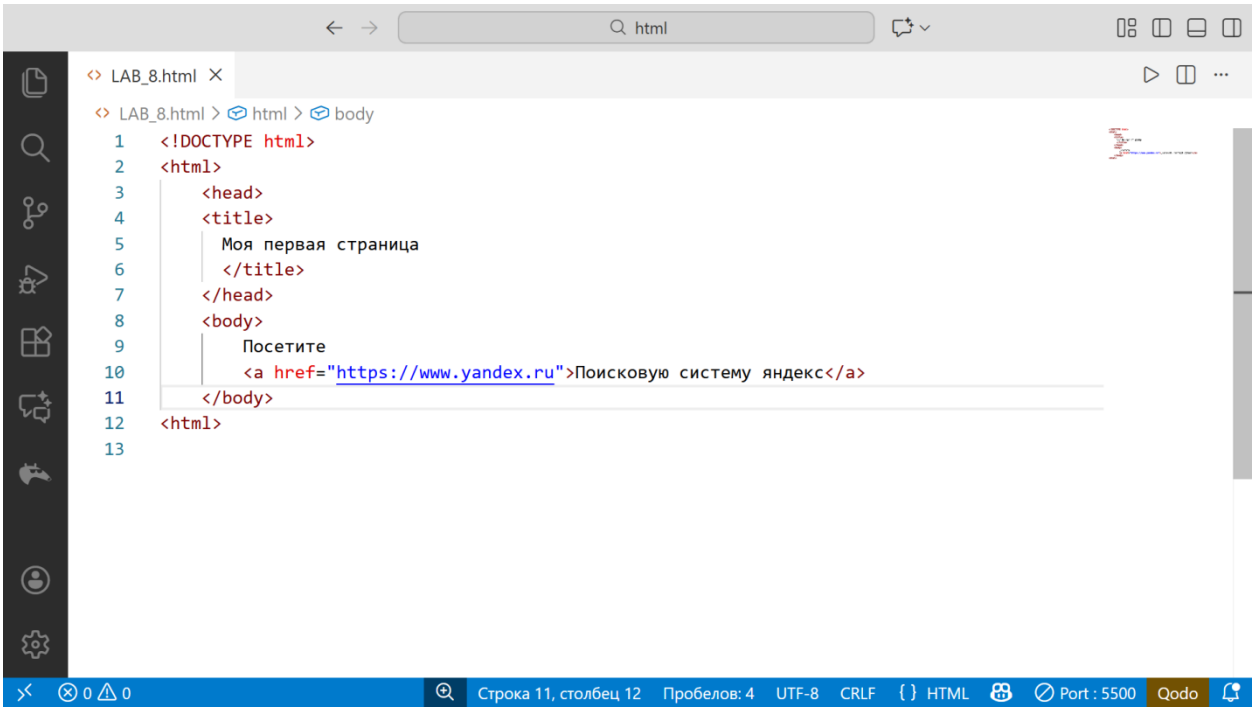



**Рисунок 8.1. Структура Html-кода с использованием тега <a> и атрибута href**

В окне браузера данная страница будет выглядеть как на рисунке 8.2.

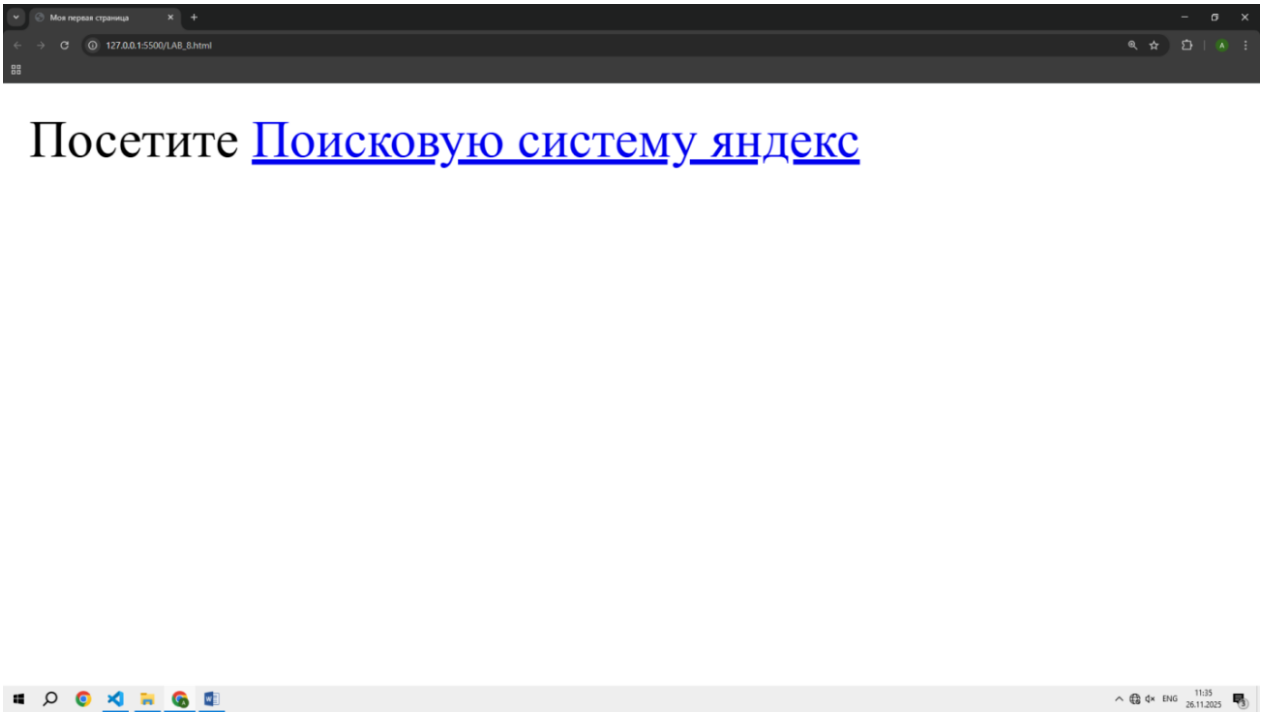



**Рисунок 8.2. Html-страница в окне браузера**

Ссылка на несуществующий файл или файл с ошибкой в значении атрибута href называется «битой», в таких случаях в окне браузера отображается сообщение с ошибкой. Текст сообщения будет различаться в разных браузерах. Также, в случае если браузер не знает тип документа, он не всегда может его открыть, в таком случае появляется окно диалога с сообщением о том, как нужно обработать документ, открыть его или сохранить в указанную папку.

Каждая ссылка на сайте может находиться в одном из трех состояний. Первое состояние характеризует еще непосещенные ссылки, по умолчанию такие ссылки выделены синим цветом и подчеркнуты.

Второе состояние - активная ссылка, является кратковременным, так как характеризует момент между нажатием ссылки на странице и открытием нового документа. Цвет такой ссылки по умолчанию красный.

Ну и третье состояние - посещенная ссылка, возникает, соответственно, как только пользователь открывает новый документ. По умолчанию такая ссылка фиолетового цвета.

В Html есть понятие относительных (внутренних) и абсолютных (внешних) ссылок.

Относительные ссылки используются, если ссылаются на какой-либо объект текущего веб-узла. При указании адреса такой ссылки не указывается протокол. В таком случае значение данного атрибута зависит от исходного расположения файлов.

Приведем пример файла, содержащего внутренние ссылки (рисунок 8.3).

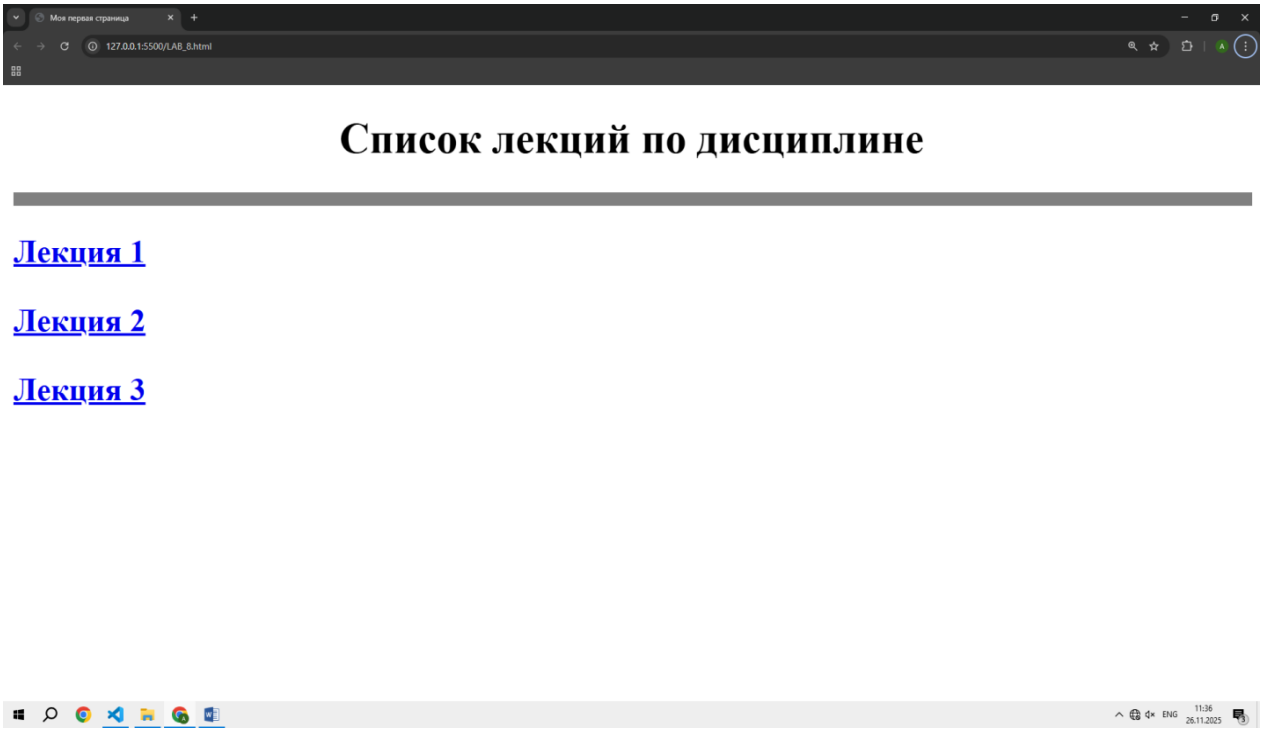



**Рисунок 8.3. Html-страница в окне браузера**

В данном примере создана ссылка из исходного документа на ссылаемый с учетом того, что они оба расположены в одной папке. Синтаксис html-кода данной страницы приведен на рисунке 8.4.

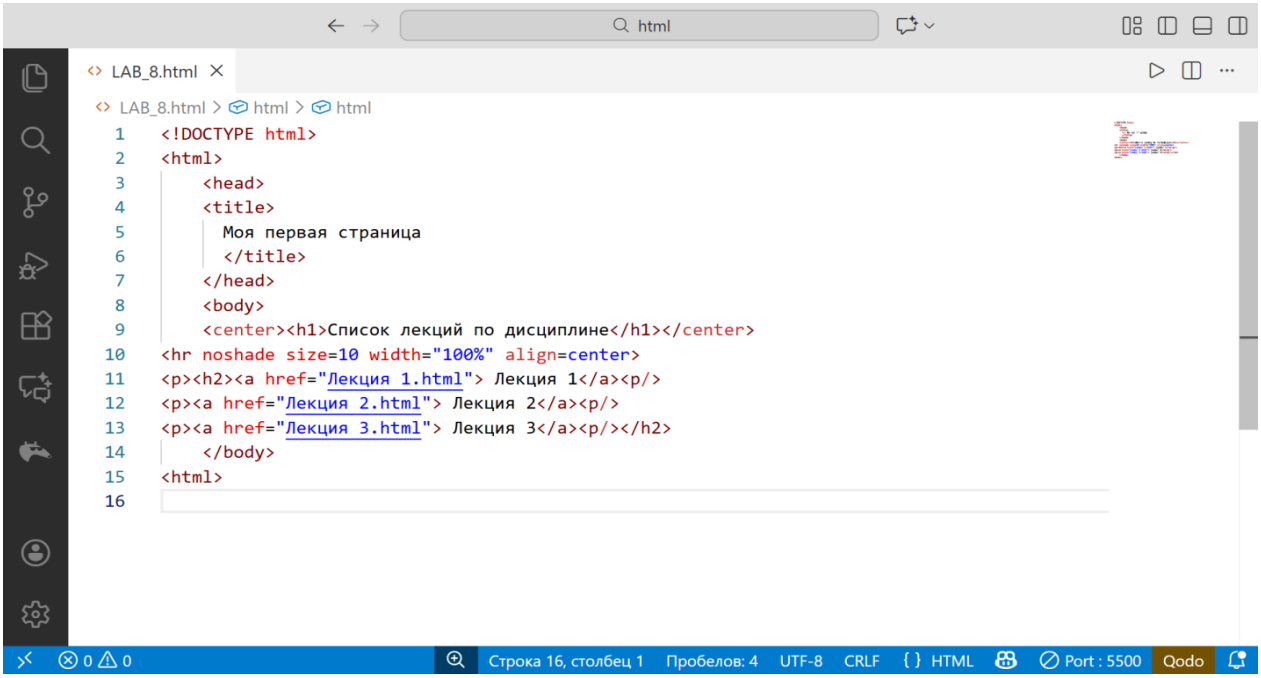



**Рисунок 8.4. Структура Html-кода с использованием относительных ссылок**

Структура папки, содержащей файлы, использованные в синтаксисе html-кода, приведена на рисунке 8.5.

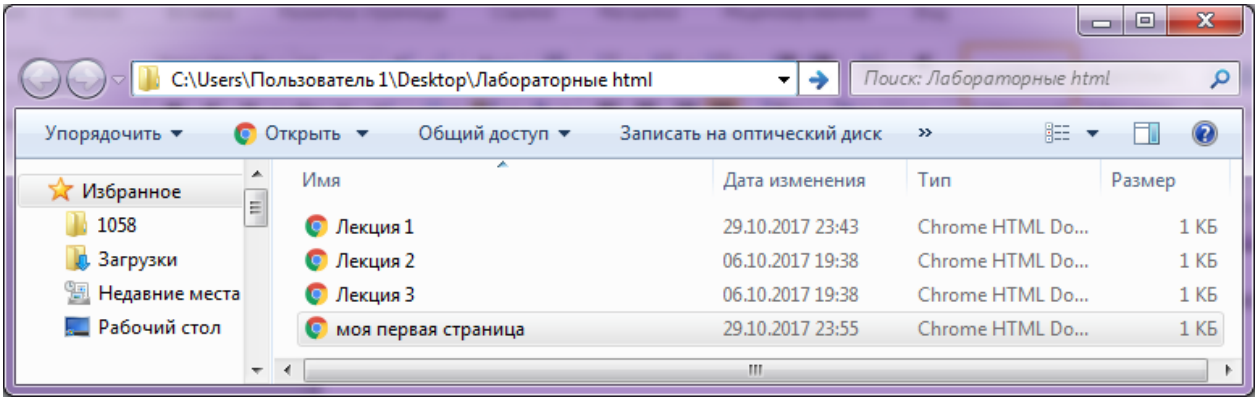



**Рисунок 8.5. Структура папки, содержащей Html-страницы**

Как видно из рисунка, наша основная страница («моя первая страница»), на которой мы размещаем ссылки, и страницы, на которые в ней ссылаются («Лекция 1, Лекция 2, Лекция 3») находятся в одной папке.

Приведем также пример, при котором эти страницы будут размещены в разных папках (рисунок 8.6).

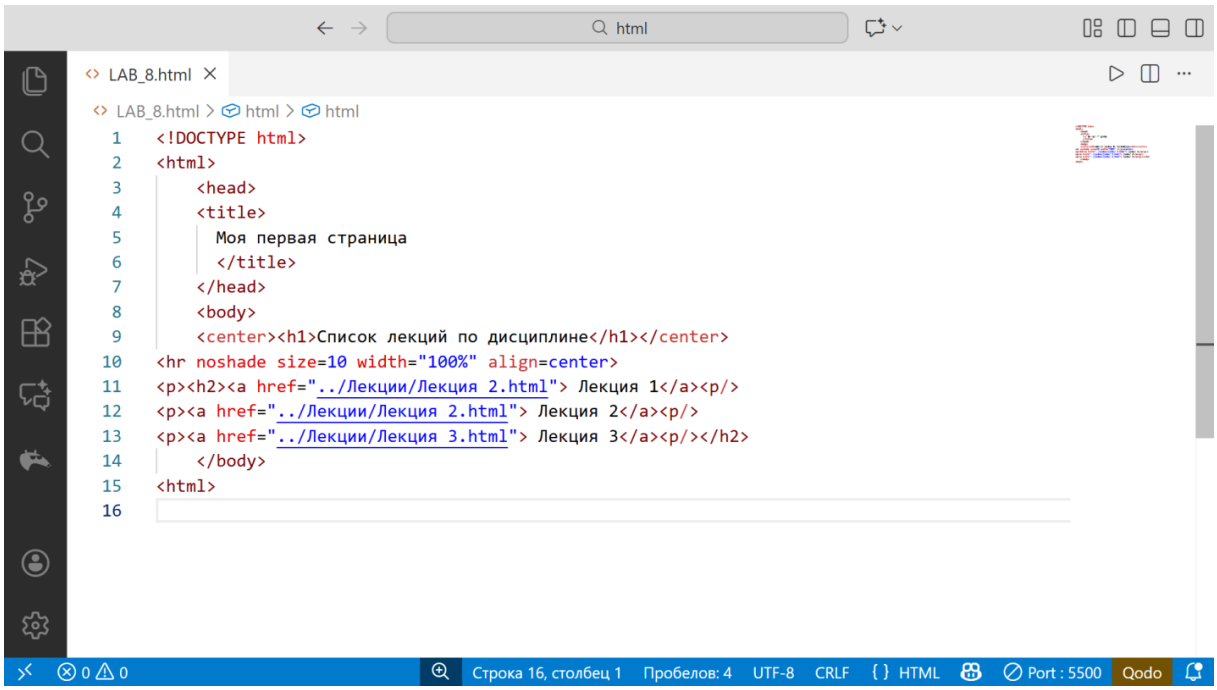



**Рисунок 8.6. Структура Html-кода**

Двоеточие в начале ссылки в данном случае означает подняться в иерархии папок на уровень выше.

Структура папок, содержащих файлы, использованные в синтаксисе html-кода, приведена на рисунке 8.7.

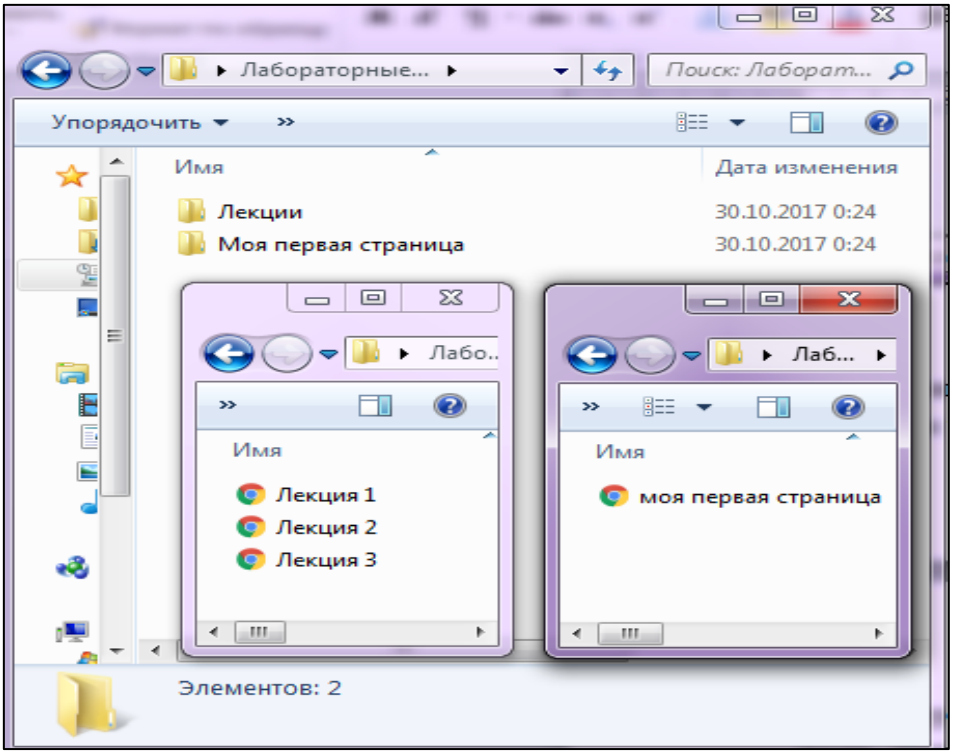



**Рисунок 8.7. Структура папок, содержащих Html-страницы**

Абсолютные ссылки указывают на страницы, не расположенные на текущем веб-узле. Поэтому в атрибуте href тега <a>необходимо указать полный URL-адрес документа, включая название протокола и адрес веб-узла.

Если файлы Лекция 1, Лекция 2, Лекция 3 разместить в каталоге C:\Users\Пользователь 1\Desktop\Лабораторные html 2, то код веб-страницы представленной на рисунке 8.3 будет выглядеть следующим образом (рисунок 8.8).

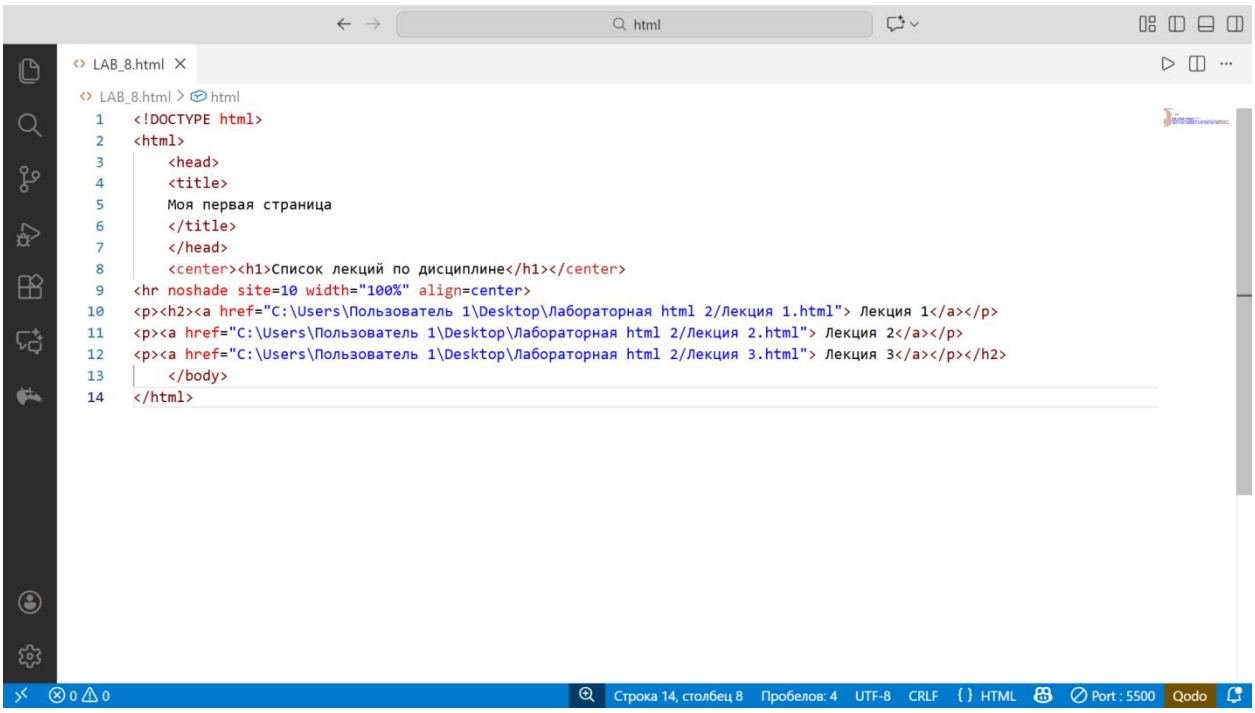



**Рисунок 8.8. Структура Html-кода с использованием внешних ссылок**

Помимо всего вышеперечисленного в Html есть возможность создания специальных якорей, которые предполагают возможность перехода от одного фрагмента внутри страницы к другому без прокрутки. При вставке якоря также используется тег <a>, однако вместо атрибута href используется атрибут name, значением которого служит имя якоря. Оно должно содержать только латинские буквы, цифры и символы, без пробелов.

Пример создания якоря в структуре html-кода приведен на рисунке 8.9.

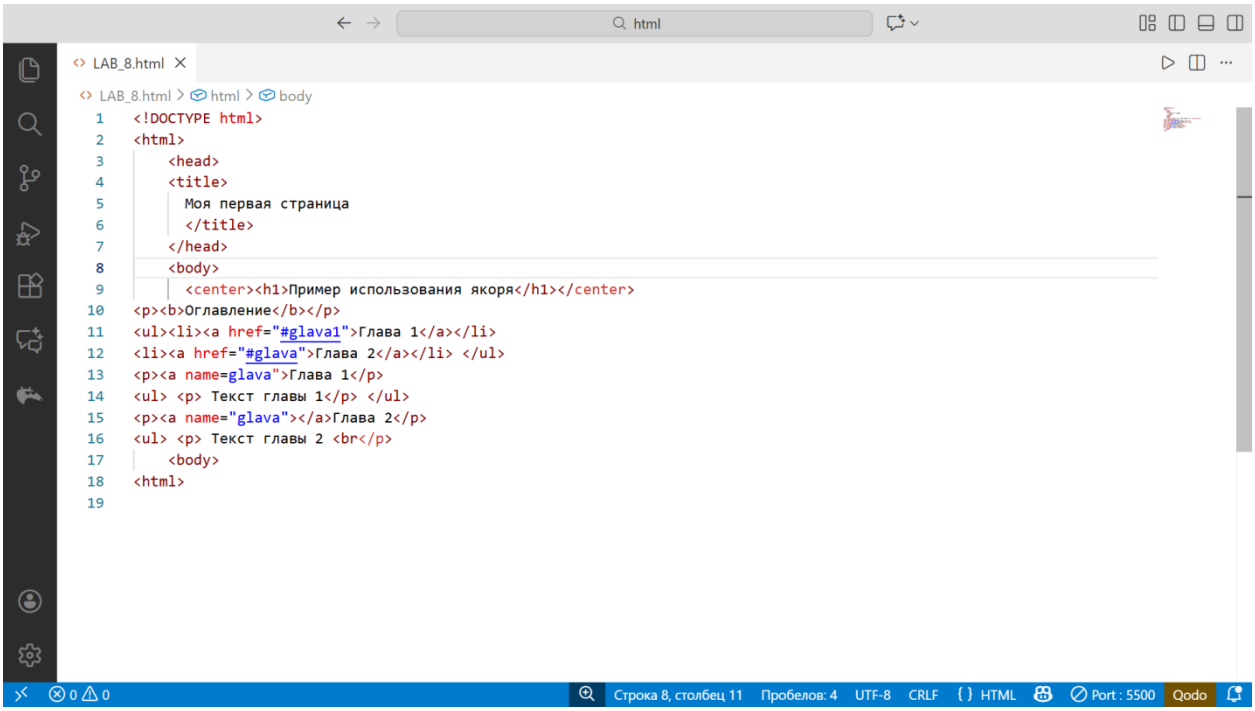



**Рисунок 8.9. Структур Html-кода с использованием якоря**

В окне браузера данная страница будет выглядеть как на рисунке 8.10.

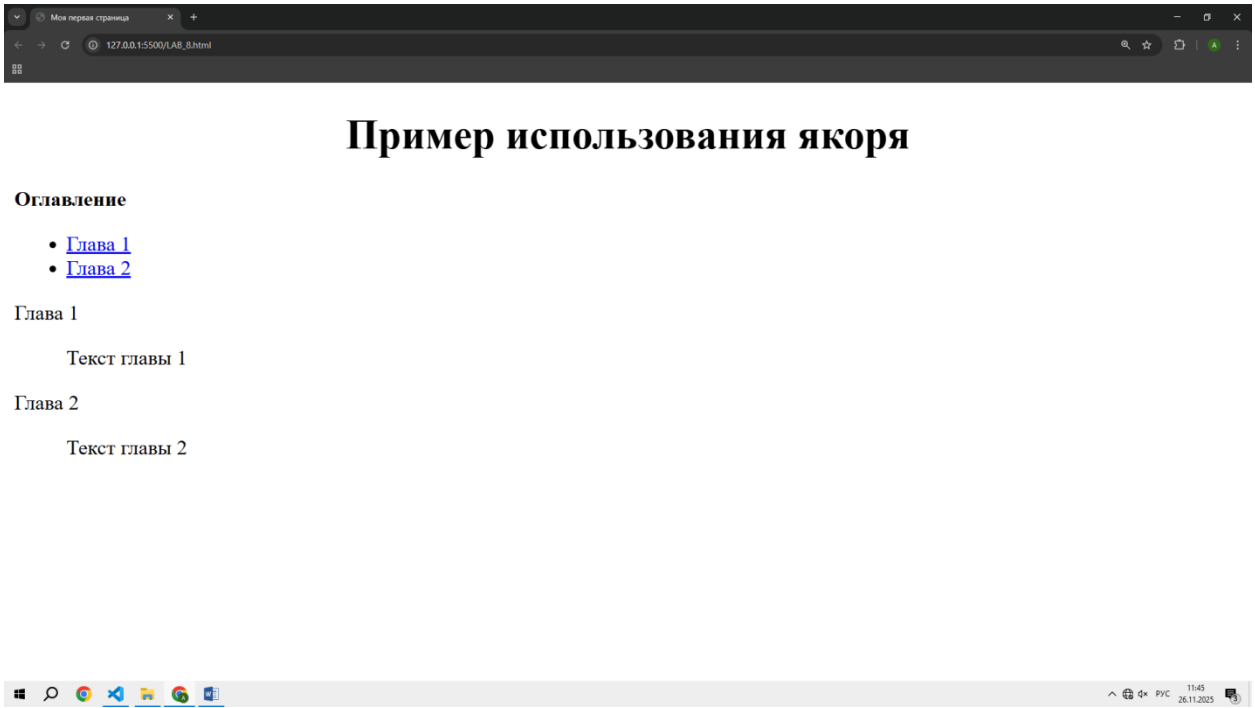



**Рисунок 8.10. Html-страница в окне браузера**

Якорь может быть вставлен как в текущий, так и в другой документ, во втором случае браузер откроет новый документ и перейдет в нем к указанному разделу.

# Лабораторная работа №9

## Работа с таблицами в HTML.

Долгое время, благодаря универсальности, а также ряду своих параметров, позволяющих управлять их видом, таблицы были стандартом верстки веб-страниц. К примеру, таблицы с невидимой границей позволяли создавать блочную структуру страниц, образуя модульную сетку. Однако, с началом применения слоев для верстки, таблицы используются для размещения табличных данных.

Как нам известно, обычная двумерная таблица состоит из строк и столбцов. Чаще всего их используют для структурирования и представления данных. Но, как мы уже сказали выше, этим возможности таблиц не ограничиваются. С помощью них мы можем сверстать макет страницы, расположив нужным образом фрагменты текста и изображений.

Основным тегом, описывающим таблицы, является тег-контейнер <table>, внутри которого должны располагаться все элементы таблицы. Строки и столбцы таблицы определяются атрибутами <tr> и <td>.

Рассмотрим ряд основных параметров тега <table>:

1. width- задает ширину таблицы;

2. bgcolor- задает цвет фона ячеек таблицы;

3. background- заливает фон таблицы рисунком;

4. border- задает рамку указанной ширины вокруг всей таблицы;

5. cellpadding- задает расстояние между текстом и внутренней границей ячейки таблицы;

6. cellspacing- задает расстояние между ячейками таблицы;

7. align- определяет выравнивание таблицы;

8. frame- задает вид рамки, обрамляющей таблицу;

9. rules- задает вид внутренних границ таблицы.

Пример html-кода с использованием тегов <tr>, <td>,<table> и его параметров приведен на рисунке 9.1.

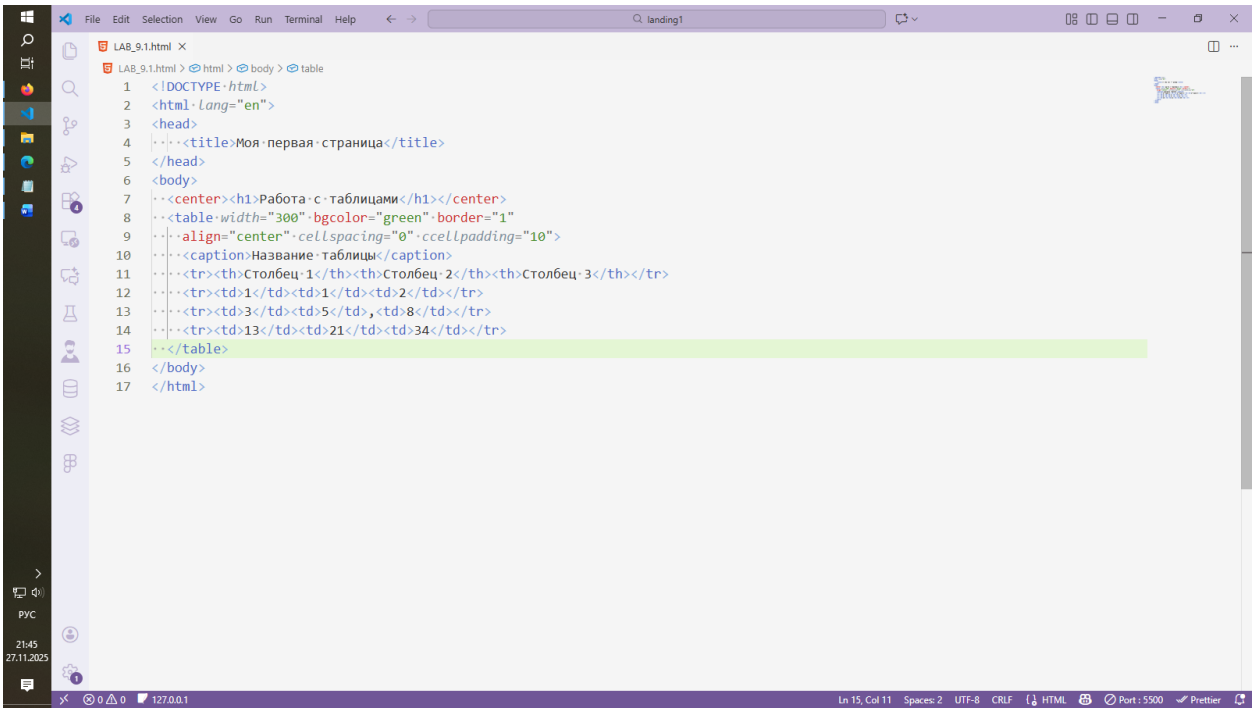



**Рисунок 9.1. Структура html-кода с использованием тегов <table>, <tr>, <td>**

В окне браузера эта страница будет выглядеть как на рисунке 9.2.

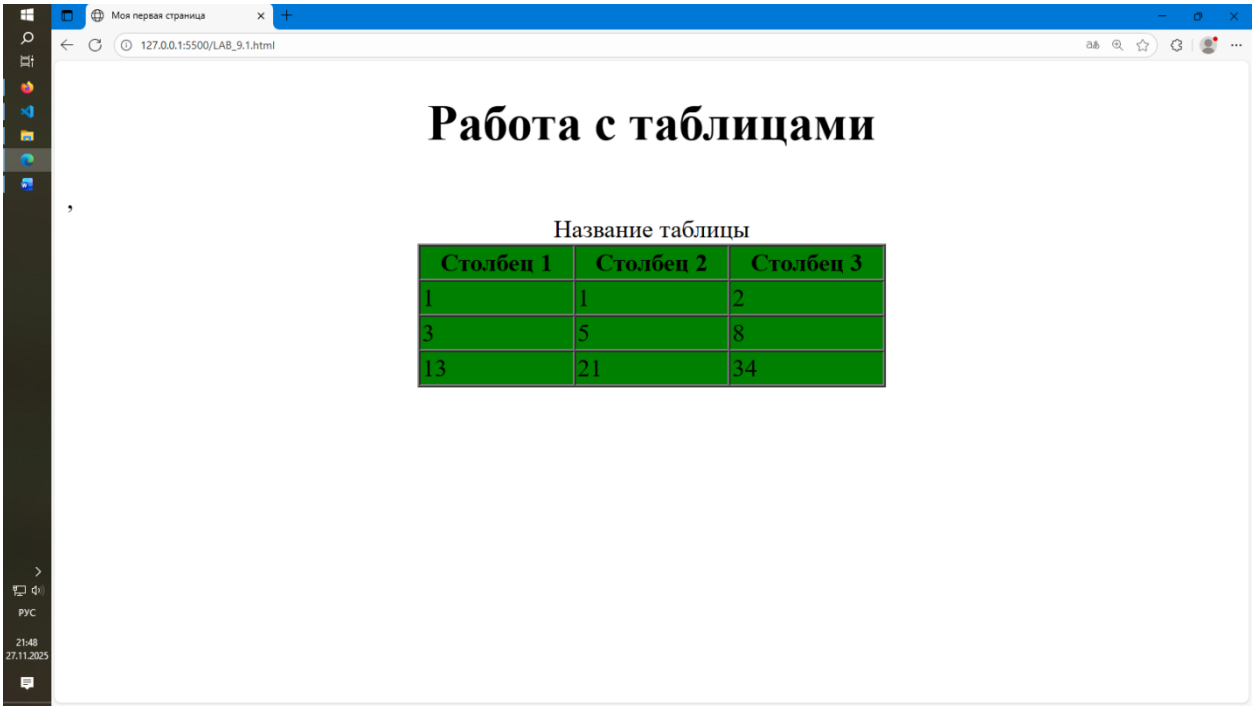



**Рисунок 9.2. Html-страница в окне браузера**

У тегов <tr> и <td> также есть свои параметры. Так как таблицы формируются построчно, заданные в теге <tr> параметры влияют на все ячейки строки (<td>). Но приписав в теге <td> определенный параметр, мы можем

повлиять на соответствующую ячейку.

Рассмотрим основные параметры:

1. align- выравнивание текста в ячейках по горизонтали;

2. valign- выравнивание текста в ячейках по горизонтали;

3. bgcolor- задание цвета строки;

4. width- задает ширину столбца;

5. height- задает высоту ячейки, при этом все ячейки данной строки будут этой же высоты.

Пример структуры html-кода с применением вышеперечисленных параметров тегов <tr>, <td> приведен на рисунке 9.3.

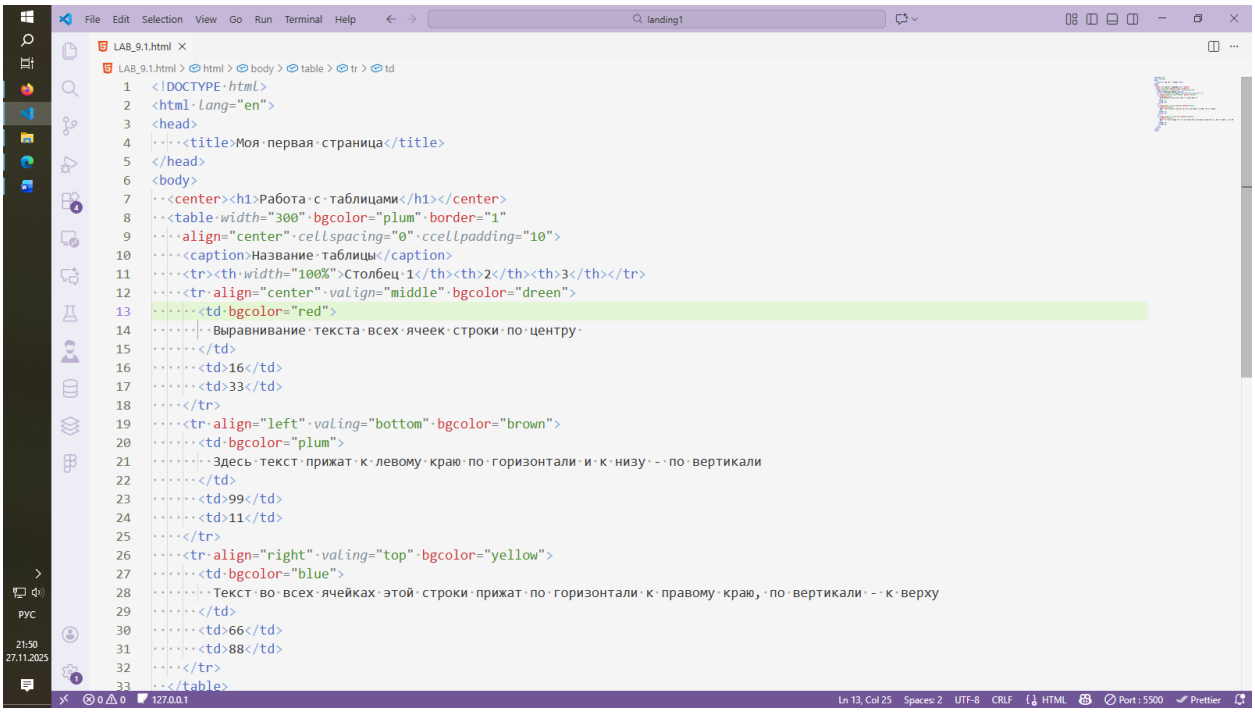



**Рисунок 9.3. Структура html-кода с использованием основных параметров тегов <tr>, <td>**

В окне браузера данная страница будет выглядеть как на рисунке 9.4.

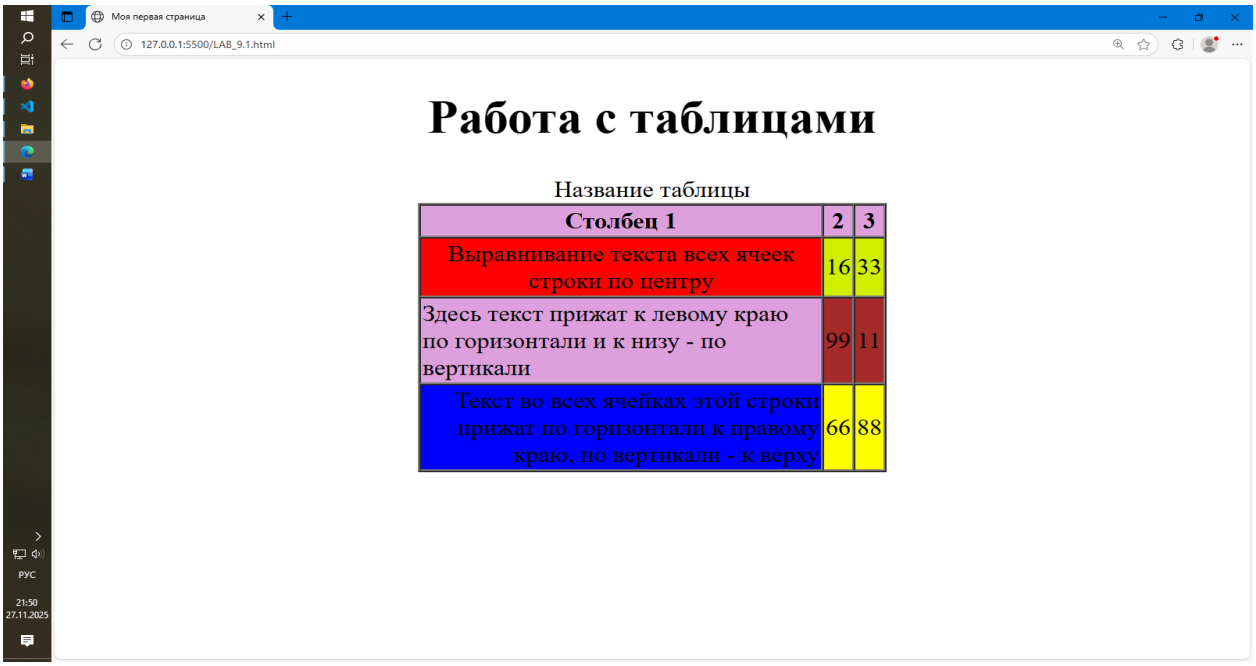



**Рисунок 9.4. Html-страница в окне браузера**

Как видно из приведенных ранее рисунков иногда вместо тега <td> используется тег <th>. Текст в ячейке, оформленной с помощью тега <th>, отображается браузером шрифтом жирного начертания и выравнивается по центру ячейки.

Пример использования тега <th> вместо тега <td> в структуре html-кода приведен на рисунке 9.5.

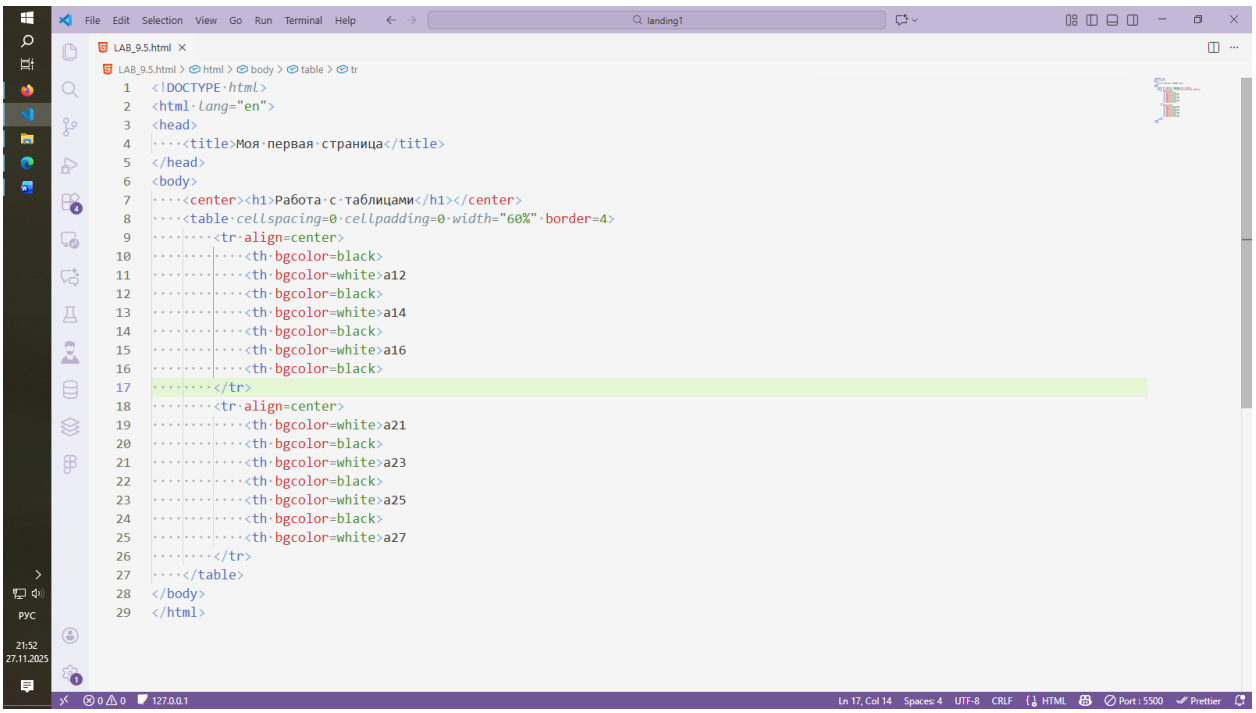



**Рисунок 9.5. Структура html-кода**

В окне браузера данная страница будет выглядеть как на рисунке 9.6.

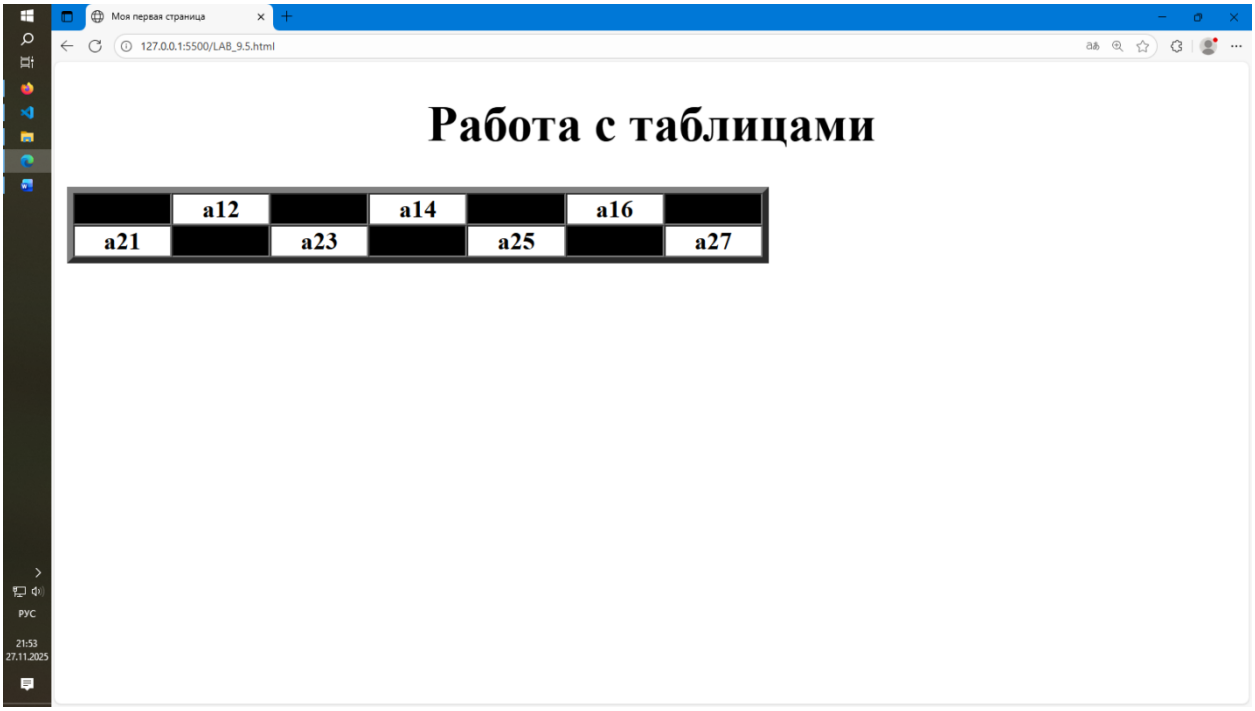



**Рисунок 9.6. Html-страница в окне браузера**

Ячейки таблиц можно объединять. Для этого используются теги атрибута <td> colspan и rowspan. Атрибут colspan объединяет ячейки по горизонтали, а rowspan- по вертикали.

Пример использования этих атрибутов в структуре html-кода приведен на рисунке 9.7.

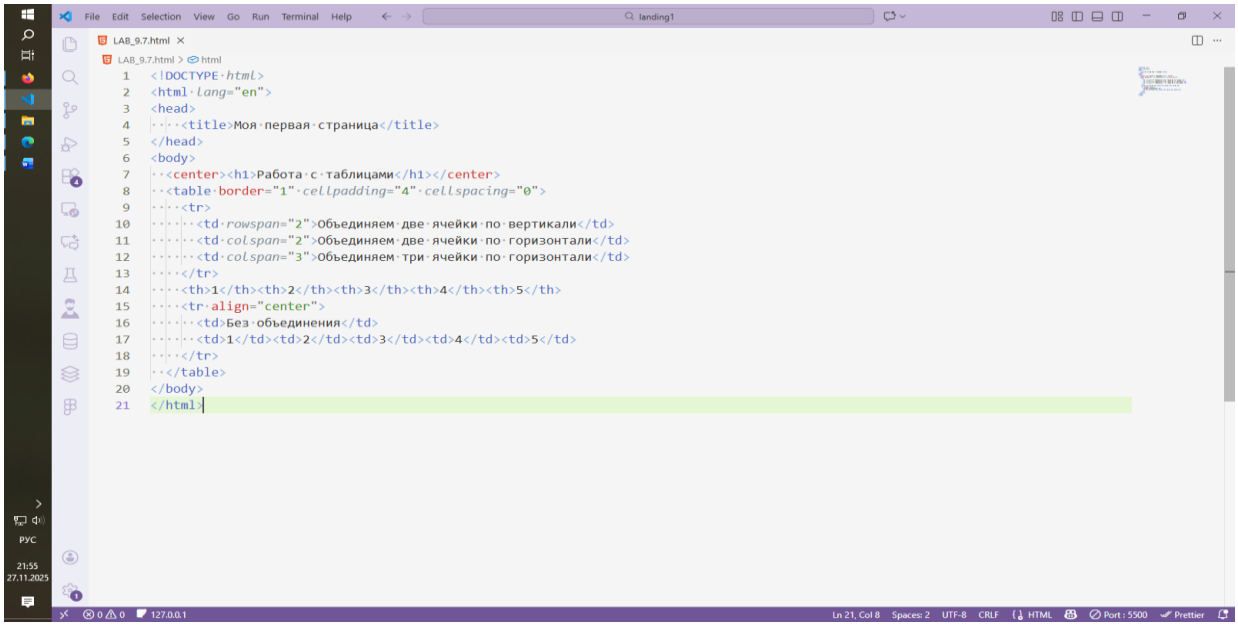



**Рисунок 9.7. Структура html-кода с использование атрибутов для объединения ячеек**

В окне браузера данная страница будет выглядеть как на рисунке 9.8.

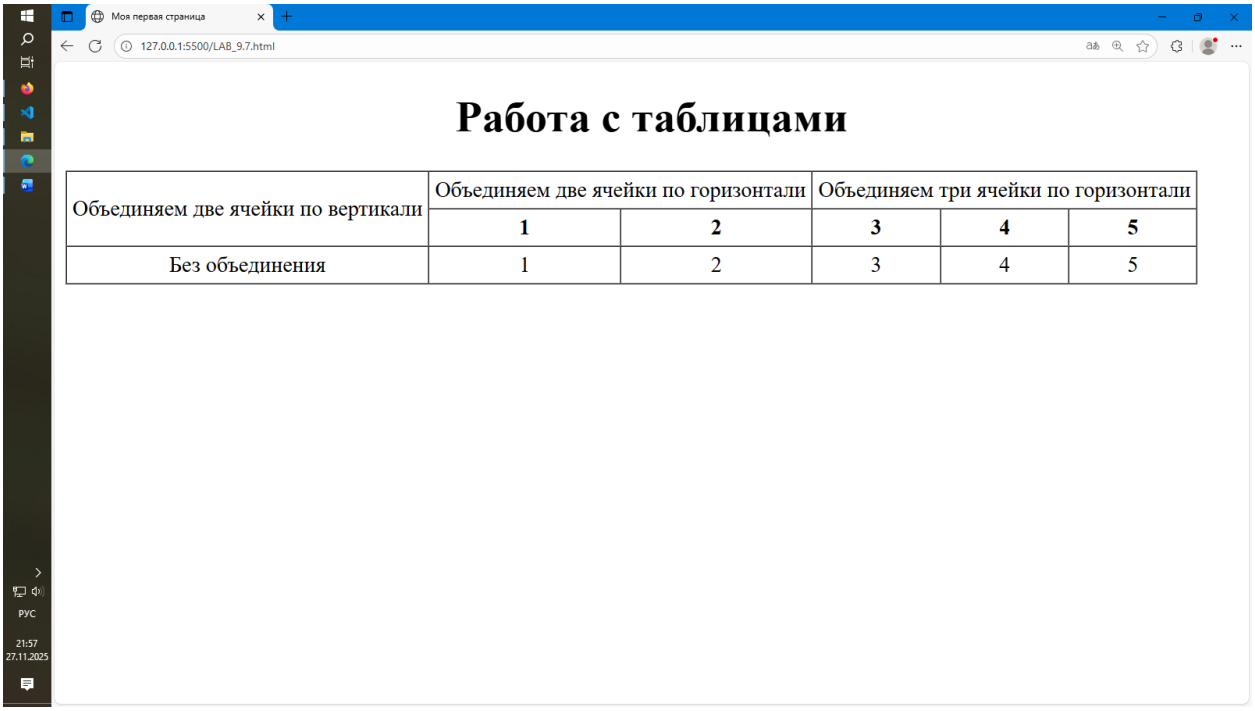



**Рисунок 9.8. Структура html-страницы в окне браузера**

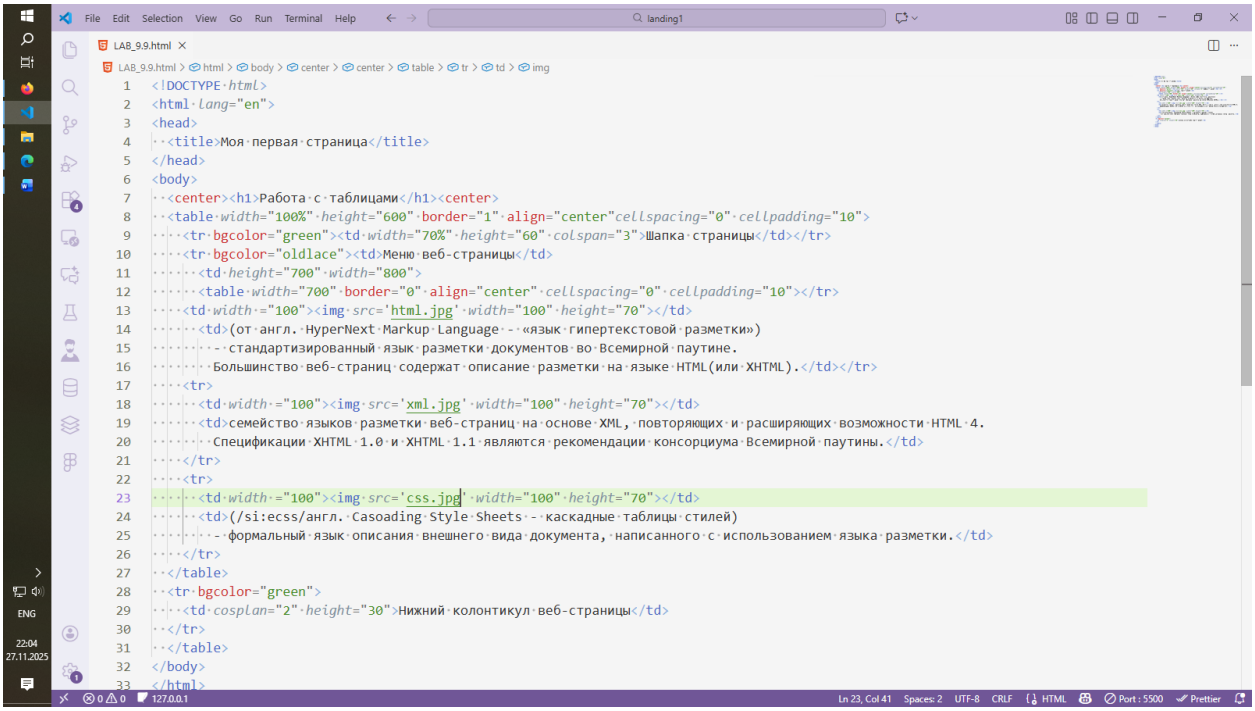
Стоит отметить, что перед использованием данных атрибутов необходимо проверить число ячеек в каждой строке, чтобы не возникло ошибок. Так, к примеру, в вышеприведенном коде мы использовали атрибут colspan два раза, первый раз объединили две ячейки, второй раз - три ячейки. Соответственно количество ячеек в следующей строке должно быть равно пяти. Если количество ячеек в каждой строке не будет совпадать, то появятся пустые ячейки.

**Примечание**
Формулировка объединение ячеек является условной, так как фактически объединения как такового не происходит ни в коде (атрибут относится к одному конкретному тегу), ни в результате (содержимое ячеек не объединяется, не заменяется, никак не взаимодействует). На самом деле ячейка, которой задан атрибут, растягивается на указанное в атрибуте количество ячеек, т.е. занимает место 2, 3 и т.д. ячеек. Соседние ячейки при этом не затрагиваются, они сдвигаются в сторону относительно растягиваемой ячейки.

У объединения ячеек есть свои недостатки, поэтому можно использовать метод задания высоты ячейки с помощью атрибута height. Но в зависимости от содержания ячеек данный атрибут может игнорироваться браузерами. Это часто выражается в таблицах, где явно задается высота ячеек и их объединение по вертикали. Для упрощения верстки применяется прием с вложенными

таблицами. Суть заключается в том, что в ячейку таблицы вкладывается другая таблица со своими параметрами.

Пример верстки страниц с использованием приема вложения таблиц приведен на рисунке 9.9.



```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Моя первая страница</title>
</head>
<body>
    <center><h1>Работа с таблицами</h1><center>
    <table width="100%" height="600" border="1" align="center" cellspacing="0" cellpadding="10">
        <tr bgcolor="green"><td width="70%" height="60" colspan="3">Шапка страницы</td></tr>
        <tr bgcolor="oldlace"><td>Меню веб-страницы</td>
            <td height="700" width="800">
                <table width="700" border="0" align="center" cellspacing="0" cellpadding="10"></tr>
                <td width="100"><img src='html.jpg' width="100" height="70"></td>
                <td>(от англ. HyperNext Markup Language - «язык гипертекстовой разметки»)
                    - стандартизированный язык разметки документов во Всемирной паутине.
                    Большинство веб-страниц содержат описание разметки на языке HTML(или XHTML).</td></tr>
                <tr>
                <td width="100"><img src='xml.jpg' width="100" height="70"></td>
                <td>семейство языков разметки веб-страниц на основе XML, повторяющих и расширяющих возможности HTML 4.
                    Спецификации XHTML 1.0 и XHTML 1.1 являются рекомендации консорциума Всемирной паутины.</td>
                </tr>
                <tr>
                <td width="100"><img src='css.jpg' width="100" height="70"></td>
                <td>(/si:ecss/англ. Casoading Style Sheets - каскадные таблицы стилей)
                    - формальный язык описания внешнего вида документа, написанного с использованием языка разметки.</td>
                </tr>
        </table>
        <tr bgcolor="green">
        <td cosplan="2" height="30">Нижний колонтикул веб-страницы</td>
        </tr>
        </table>
</body>
</html>
```

**Рисунок 9.9. Структура Html-кода с использованием вложения таблиц**

В окне браузера данная страница будет выглядеть как на рисунке 9.10.

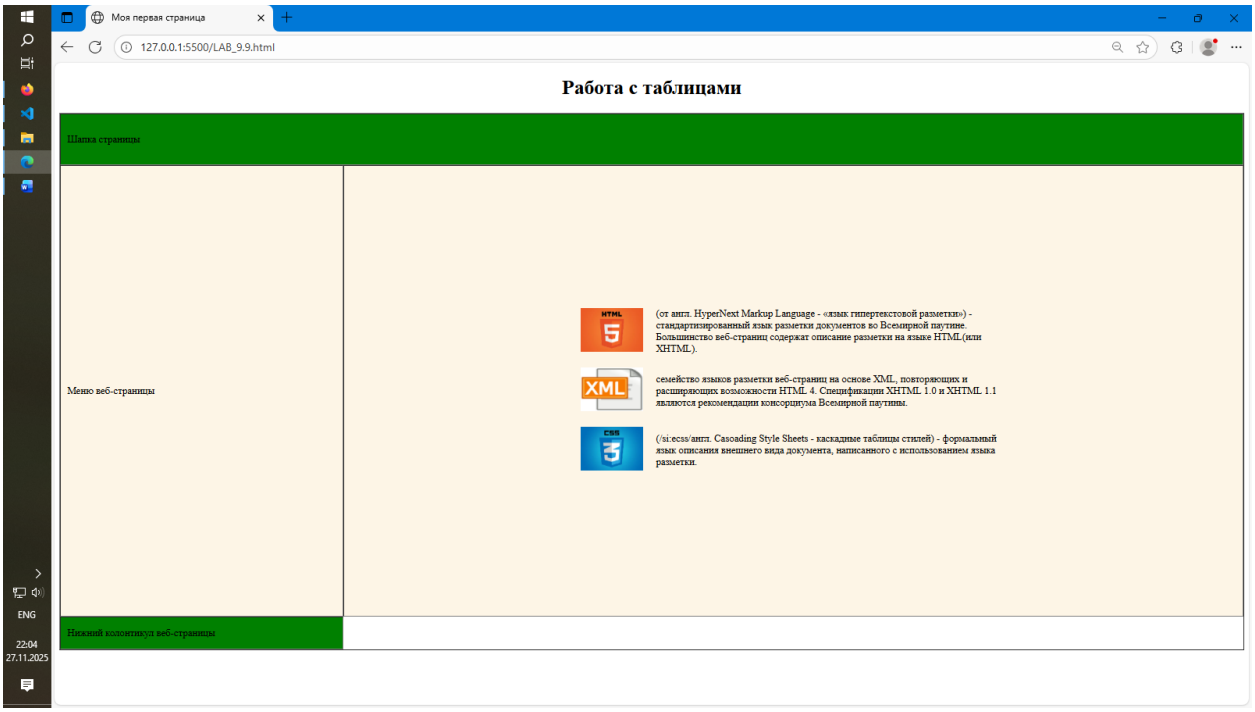
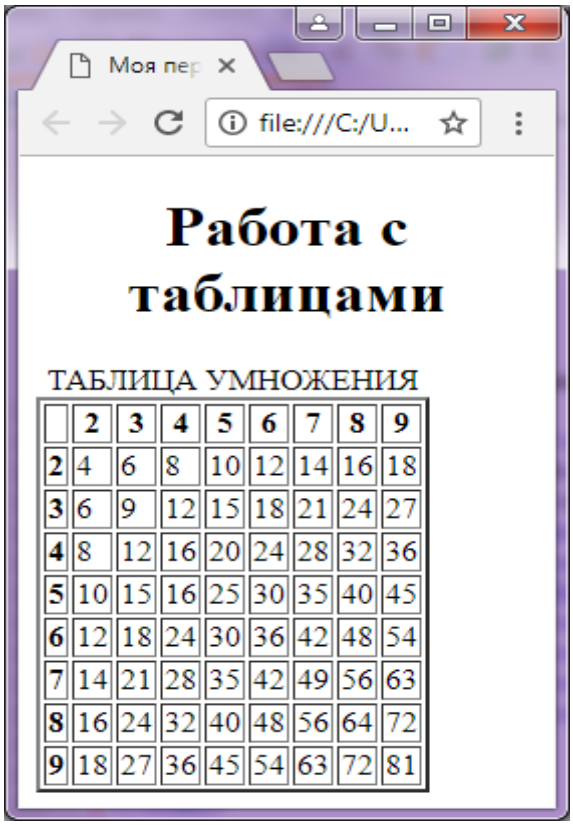



**Рисунок 9.10. Html-страница в окне браузера**

Для закрепления материала из лабораторной работы выполните следующие самостоятельные задания:

1. Постройте таблицу умножения Пифагора как на рисунке 9.11.




| | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|
| 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 | 18 |
| 3 | 6 | 9 | 12 | 15 | 18 | 21 | 24 | 27 |
| 4 | 8 | 12 | 16 | 20 | 24 | 28 | 32 | 36 |
| 5 | 10 | 15 | 16 | 25 | 30 | 35 | 40 | 45 |
| 6 | 12 | 18 | 24 | 30 | 36 | 42 | 48 | 54 |
| 7 | 14 | 21 | 28 | 35 | 42 | 49 | 56 | 63 |
| 8 | 16 | 24 | 32 | 40 | 48 | 56 | 64 | 72 |
| 9 | 18 | 27 | 36 | 45 | 54 | 63 | 72 | 81 |

**Рисунок 9.11. Задание на самостоятельное выполнение (таблица умножения Пифагора)**

2. Постройте таблицу с объединенными ячейками как на рисунке 9.12.

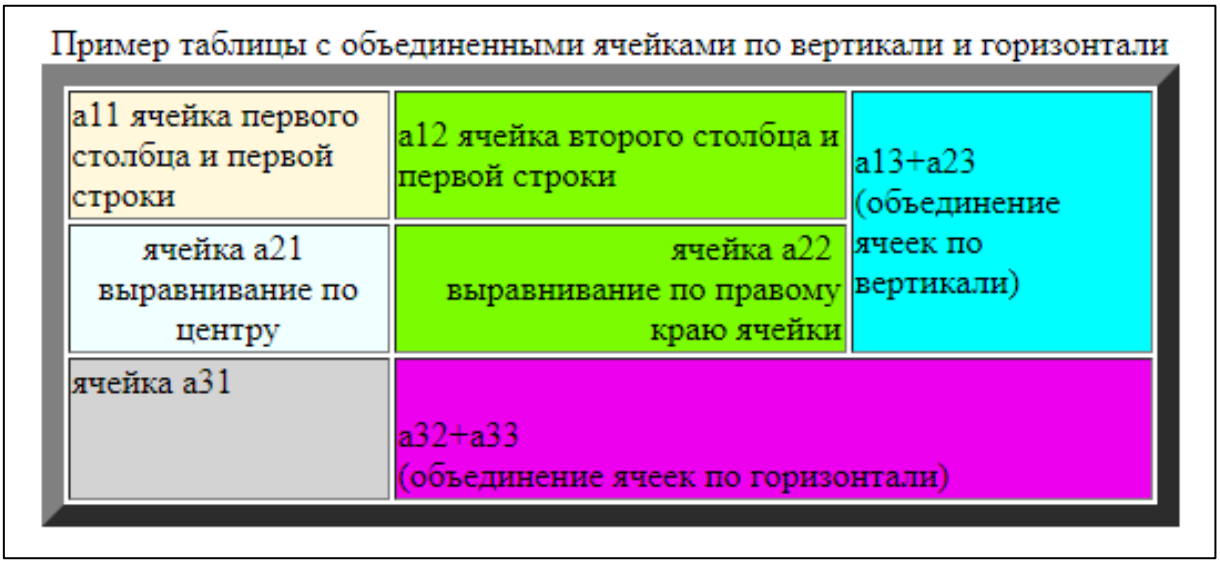



**Рисунок 9.12. Задание на самостоятельное выполнение (таблица с объединенными ячейками)**

# РАЗДЕЛ II. ВВЕДЕНИЕ В CSS

## Лабораторная работа № 1

CSS (Cascading Style Sheets, каскадные таблицы стилей) - набор параметров форматирования, который применяется к элементам документа (HTML тегам) для изменения их внешнего вида. Они могут применяться как к отдельному тегу, так и одновременно ко всем идентичным элементам на всех страницах сайта. По сути CSSявляется дополнением к HTML, которое расширяет его возможности, так как HTMLимеет ограниченный набор инструментов, который не позволяет воплотить все дизайнерские и функциональные задумки разработчика. Кроме того CSSпозволяет разгрузить HTML документ, так как позволяет «вынести» повторяющиеся стилевые описания в отдельный файл, тем самым экономится объем, трафик, время и т.д., в результате чего HTML документ становится легким, удобным для чтения и редакции.

Внедрить CSS в документ HTML, то есть связать стилевое описание тега непосредственно с самим тегом, можно тремя способами. Первый способ подходит только если есть один элемент в HTML-документе, нуждающийся в отдельном стилевом описании, при этом стилевое описание производится непосредственно в самом элементе. Второй способ пригоден, если стиль страницы принципиально отличается от общего дизайна сайта, при этом необходимо написать стилевое описание для всех идентичных элементов HTML документа. Третий способ предполагает вынесение стилевого описания элементов HTML в отдельный файл CSS. Данный способ является наиболее эффективным использованием таблицы каскадных стилей.

Рассмотрим подробнее каждый вариант и ознакомимся с синтаксисом CSS.

### Первый подход: атрибут style

Практически каждый HTML тег имеет атрибут style, предполагающий применение стилевого описания к данному тегу. Значение данного атрибута и будет стилевым описанием для элемента.

Пример HTML кода с применением атрибута style приведен на рисунке 1.1.

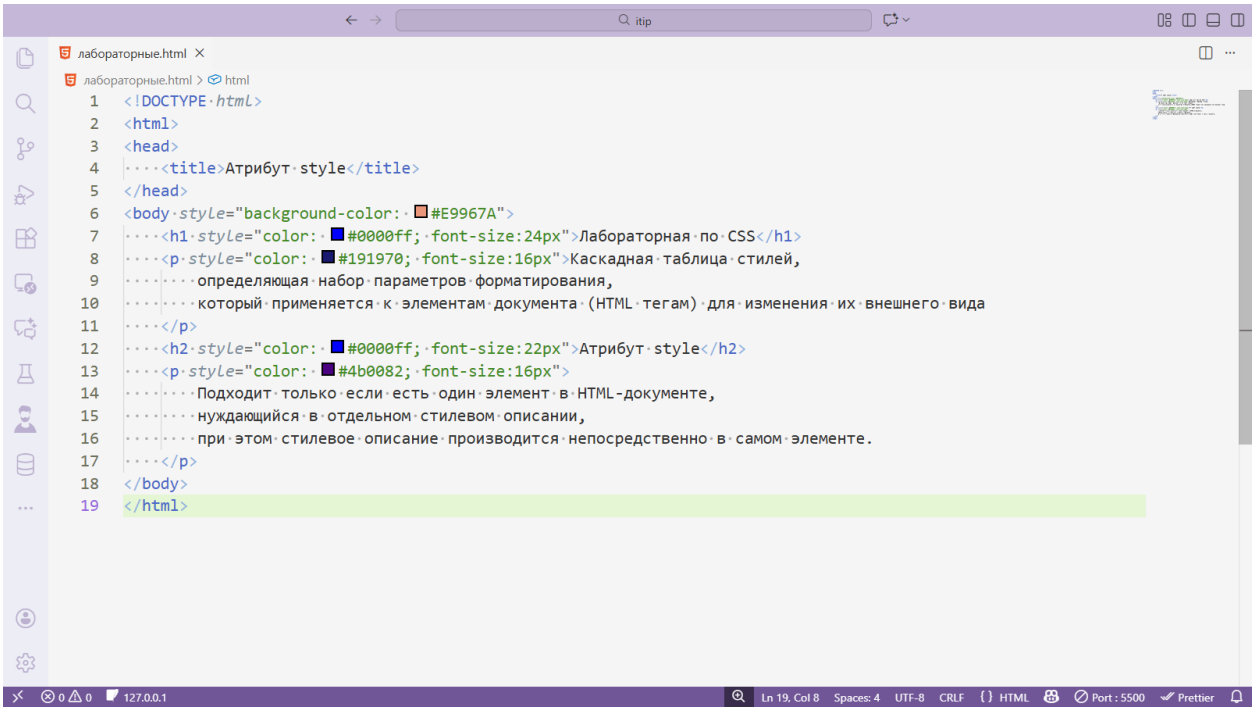



**Рисунок 1.1. Пример HTML-кода с использованием атрибута style**

В окне браузера данная страница будет выглядеть как на рисунке 1.2.

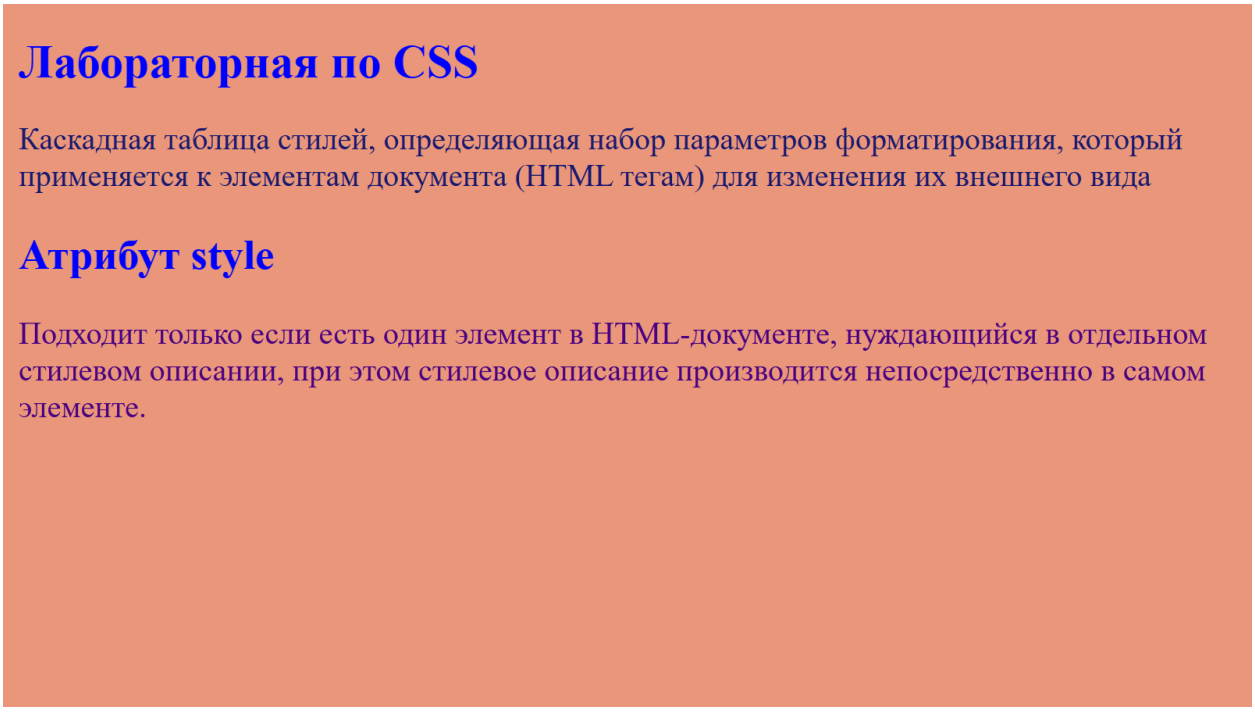



**Рисунок 1.2. HTML-страница в окне браузера**

Уточним еще раз, что данный способ внедрения CSS подходит только при малом количестве элементов, которым необходимо задать определенный стиль.

**Второй подход: тег <style>.**

Для того чтобы описать необходимые элементы одновременно на всей странице в заголовок HTML документа внедряют парный тег <style></style>, в котором описываются нужные нам элементы. Тег <style>необходимо внедрять в заголовок HTML документа, между тегами <head></head>.

У тега <style>есть атрибут type, который сообщает браузеру какой синтаксис использовать для правильной интерпретации стилей. Данный атрибут принимает значение одного из MIME типов данных. MME тип данных - спецификация для передачи по сети файлов различного типа: изображений, музыки, текстов, видео и т.д. Указание MME-типа используется в HTMLобычно при передаче данных форм и вставки на страницу различных объектов. Для правильной интерпретации браузерами CSS значение type(MME-тип данных) должно равняться text/css.

Пример HTML-кода с использованием тега <style>приведен на рисунке 1.3.

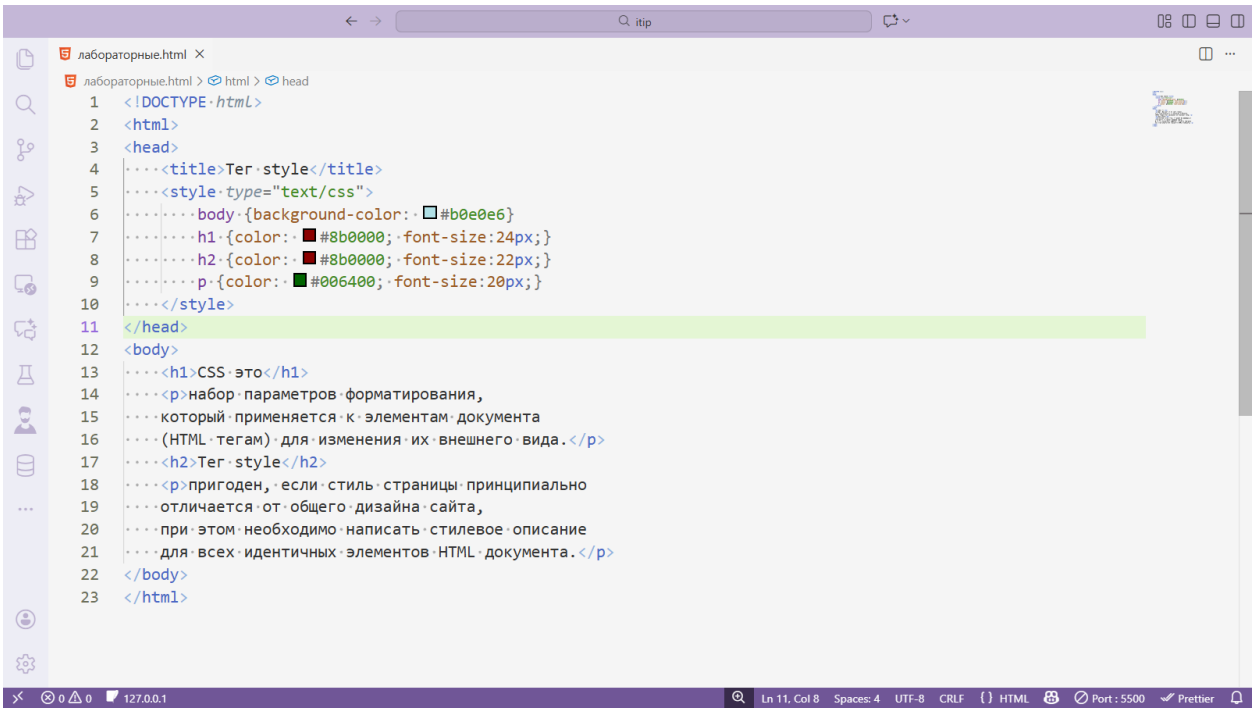



**Рисунок 1.3. Пример HTML-кода с использованием тега<style>**

Как видно из рисунка при данном подходе внедрения CSS стилей мы не прописываем к каждому элементу стиль индивидуально, а выносим его в за-

головок документа. При этом сам документ стал меньше, так как мы удалили все повторяющиеся стилевые описания для каждого отдельно взятого элемента. Отметим также, что если в блоке объявления стилей элементов (фигурные скобки) указывается несколько свойств элемента, то они между собой разделяются точкой с запятой.

В окне браузера HTML-страница приведенная на рисунке 1.3 будет выглядеть как на рисунке 1.4.

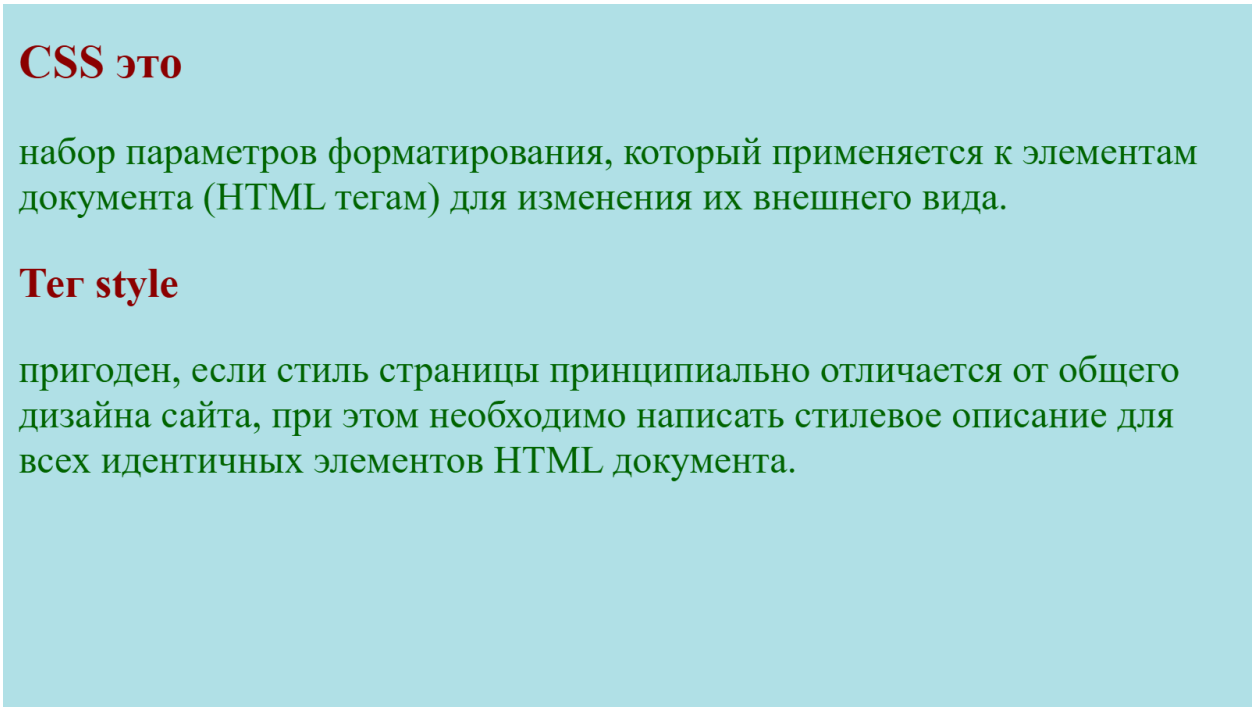



**Рисунок 1.4. HTML-страница в окне браузера**

### Третий подход: CSS в отдельном внешнем файле

Как мы выше уже говорили, данный способ, при котором мы выносим во внешний файл все данные, касающиеся дизайна сайта, является наиболее эффективным.

Для начала создайте в Visual Studio Code страницу с кодом как на рисунке 1.5.

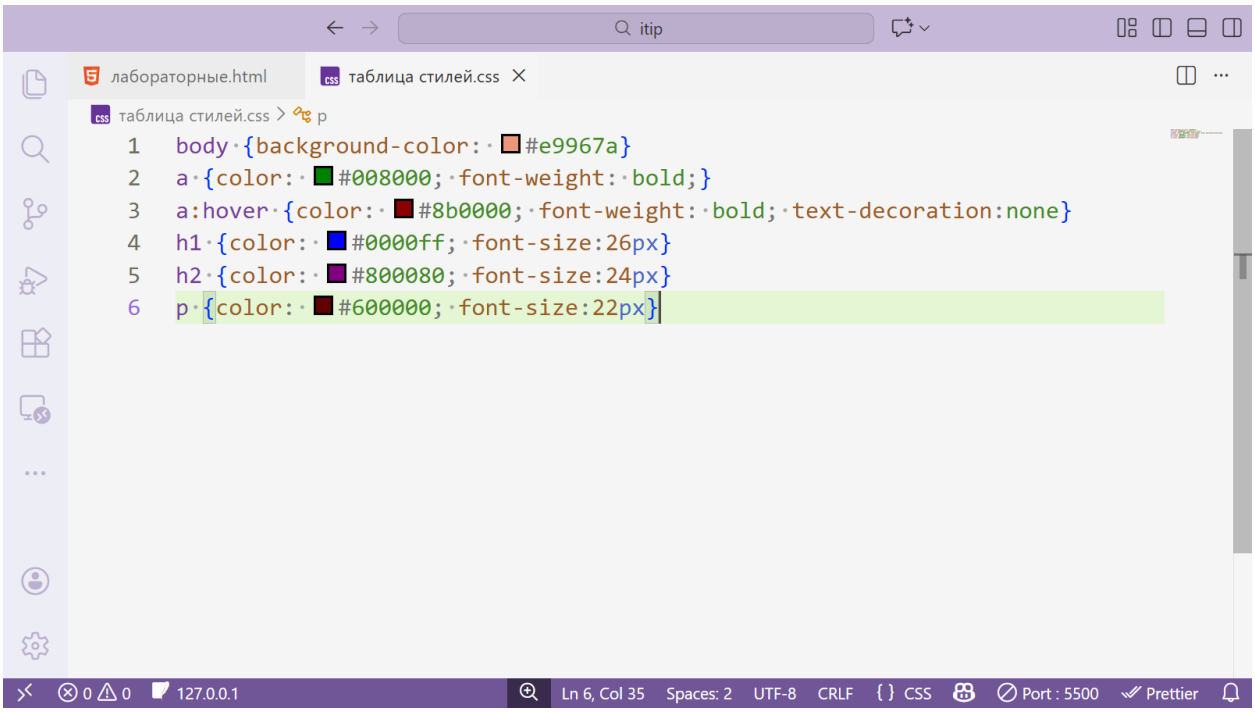



**Рисунок 1.5. Страница с кодом для CSS файла**

Сохраняем данный файл с расширением *.css. Теперь у нас имеется свой файл со стилевым описанием. Необходимо его связать с HTML. Для этого нам нужен тег <link>(связь). Данный тег является многоцелевым и служит для связывания HTML документа с внешними файлами, которые обеспечивают его должную работу. Тег <link>является ссылкой, только не для пользователей, а для программ-обозревателей - браузеров. Он прописывается в заголовке документа между тегами <head></head>.

У тега <link>есть два атрибута:

1. href- прописываем путь к файлу со стилями;
2. rel- определяем отношения между текущим документом и файлом, на который дается ссылка, в случае подключения таблицы стилей значением данного атрибута прописываем stylesheet.

Пример HTML-кода с подключением CSS как внешнего файла приведен на рисунке 1.6.

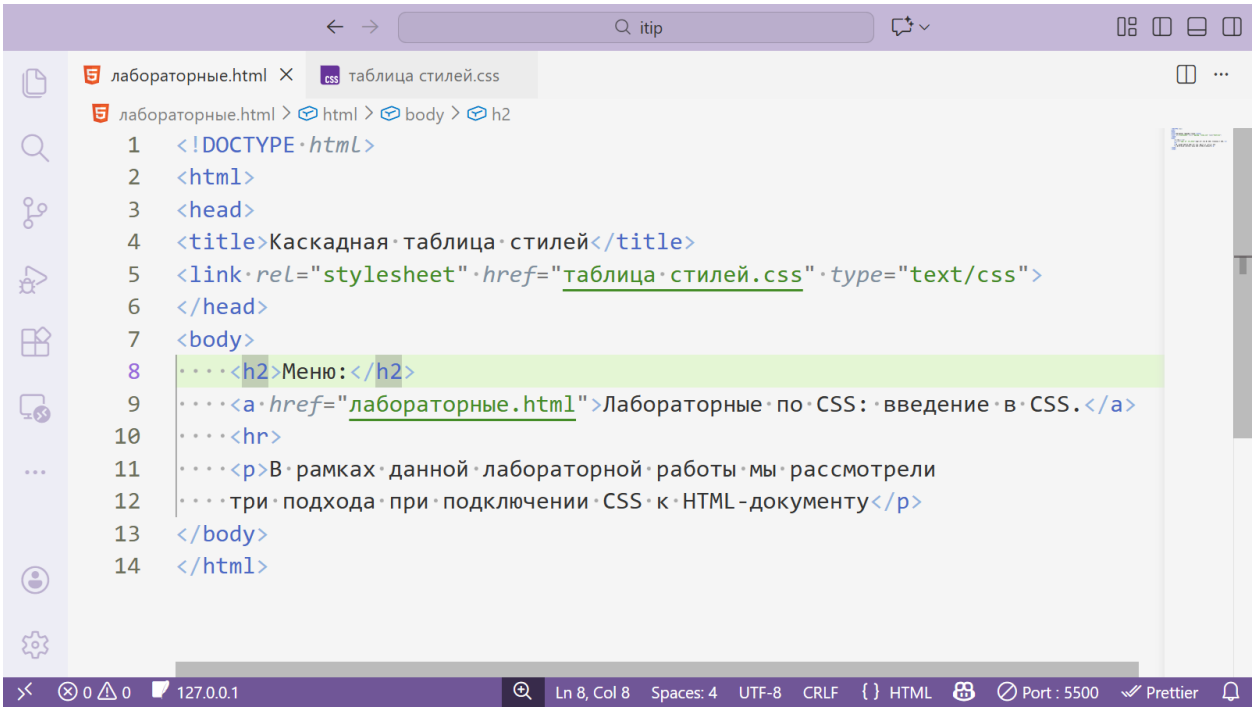



**Рисунок 1.6. Пример кода HTML-страницы с использованием ссылки для подключения внешнего файла CSS**

В окне браузера данная страница будет выглядеть как на рисунке 1.7.

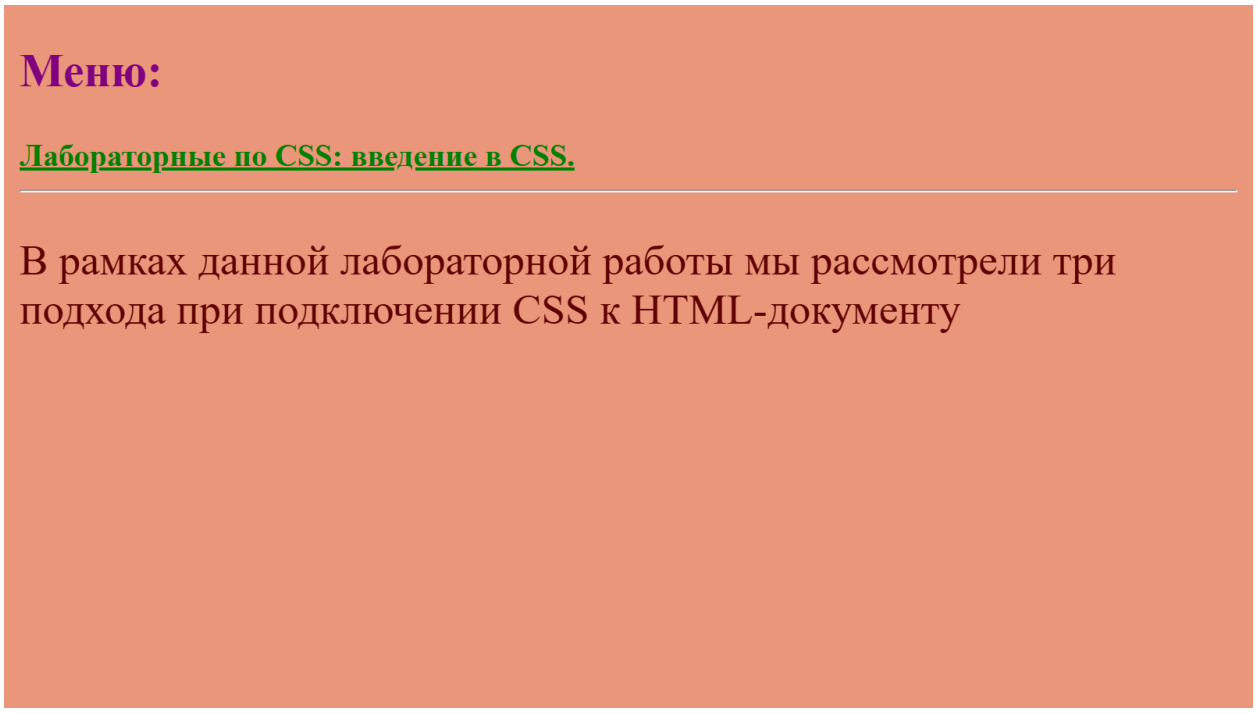



**Рисунок 1.7. HTML-страница в окне браузера**

# Лабораторная работа №2

## Свойства текста в CSS

Свойства текста позволяют задавать параметры слов и предложений, а также их взаимное расположение.

### Выравнивание текста

В HTML выравнивание текста регулировалось атрибутом align.

В CSS для этого используется свойство text-align. Данное свойство выравнивает текстовое содержание страницы относительно окна браузера или же относительно элемента родителя, к примеру, относительно блока div.

Text-align имеет следующие значения:

1. Left- выравнивает текст по левому краю;
2. Right- выравнивает текст по правому краю;
3. Center - выравнивает текст по центру;
4. Justify- выравнивает текст по обоим краям.

Пример синтаксиса кода с использованием данного свойства выглядит следующим образом:

**<pstyle="text-align: center">Выравнивание текста по центру </p>**

В данном случае свойство CSS внедряется непосредственно в HTML тег. Пример внедрения тега style в заголовок документа приведен на рисунке 2.1.

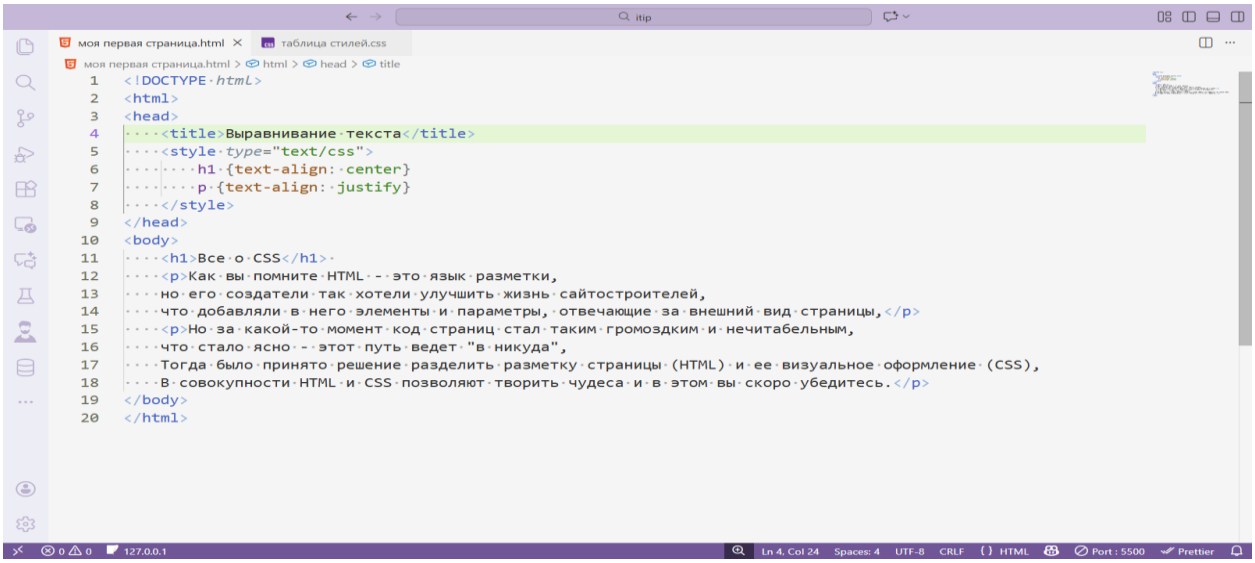



**Рисунок 2.1 Пример использования тега style со свойством text-align в заголовке HTML-документа**

В окне браузера данная страница будет выглядеть как на рисунке 2.2.

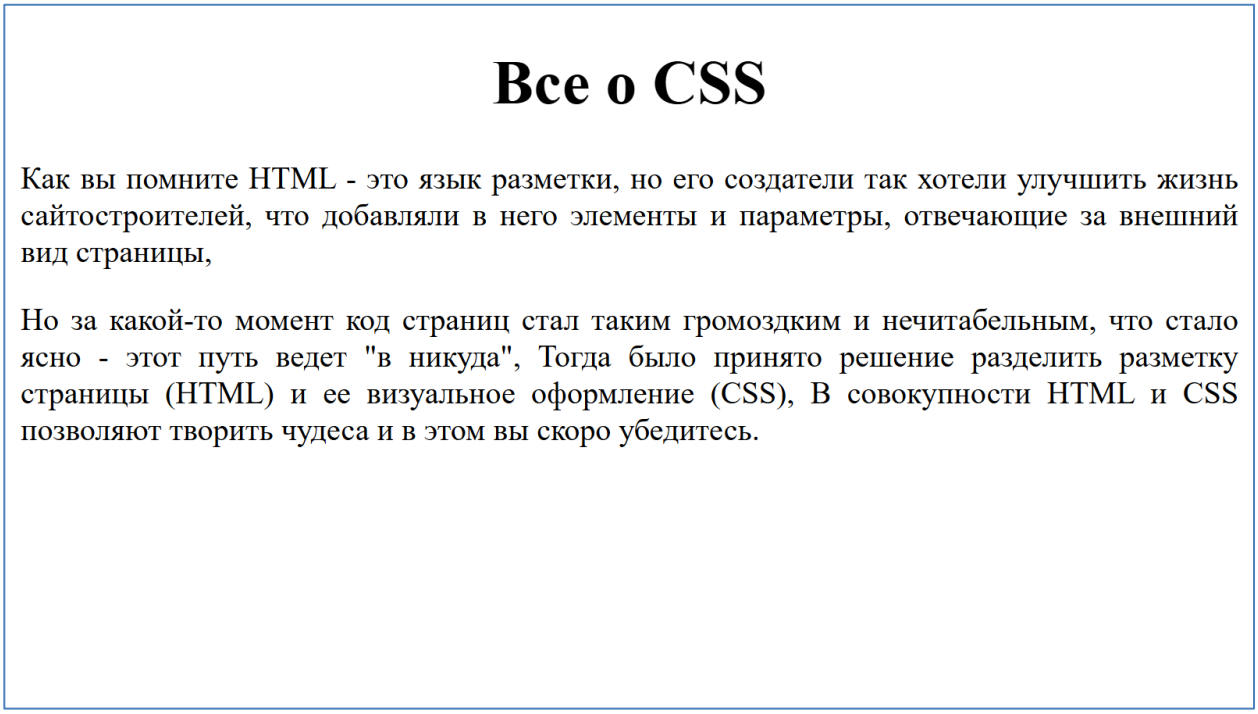



**Рисунок 2.2. HTML-страница в окне браузера**

### Оформление текста

Свойство text-decoration отвечает за оформление текста, используя одно или несколько из перечисленных ниже значений:

1. Blink- текст будет мигать.
2. Line- through - делает текст перечеркнутым.
3. Overline - надчеркивание текста.
4. Underline - подчеркивание текста.
5. None - текст без оформления.

Пример синтаксиса данного свойства приведен на рисунке 3.

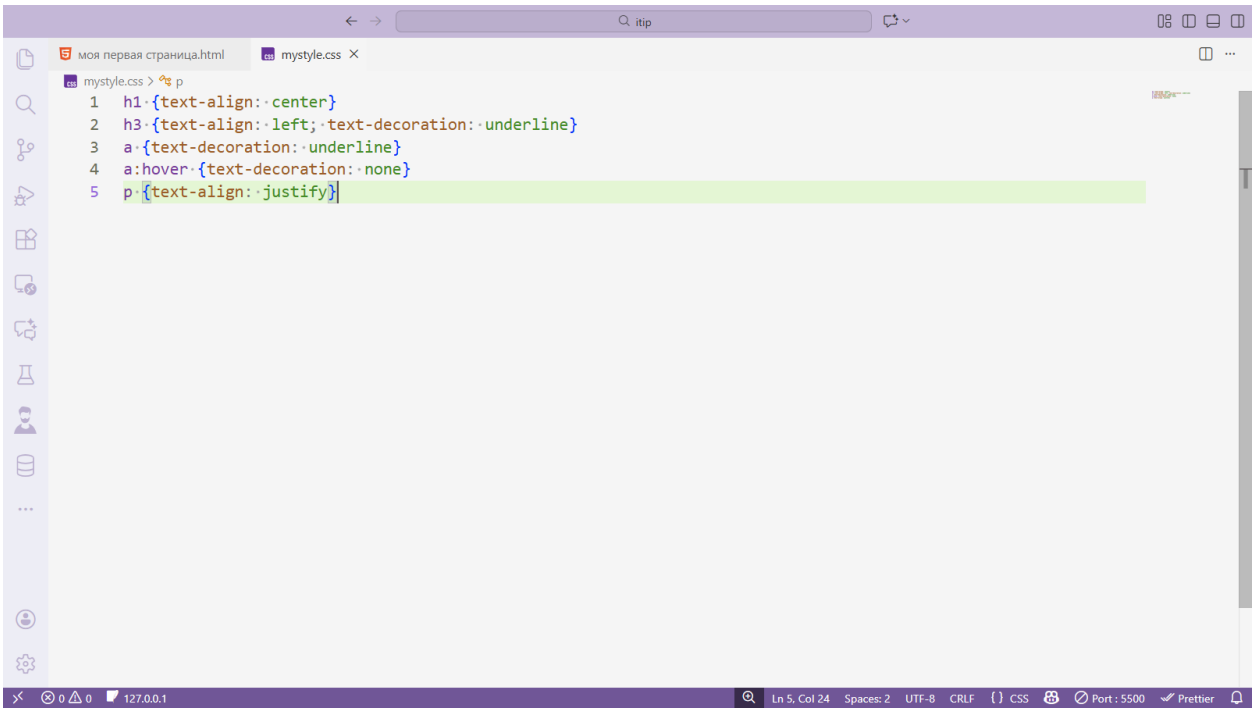



**Рисунок 2.3. Синтаксис CSS-кода с использованием свойства text-decoration**

В CSS существуют четыре псевдокласса, они позволяют работать со ссылками. У ссылок есть четыре состояния: простая, активная, посещенная и та, на которую наведен курсор. Состояние ссылок зависит от действия пользователя, и браузер, в зависимости от этих действий может применять разные стили. Для описания этих стилей и существуют следующие псевдоклассы.

1. a:link - задает стиль обычной ссылки;

2. a:active - задает стиль активной ссылки;

3. a:visited - задает стиль посещенной ссылки;

4. a:hover - задает стиль ссылки, на которую наведен курсор.

Как видно из рисунка 2.3 в данном внешнем файле CSS был декорирован тег-ссылка **a**.

В первом случае было использовано свойство text-decoration со значением underline. Во втором случае использован псевдокласс hover, в котором запрещено подчеркивание. Данный псевдокласс создает эффект динамичности.

Пример подключения данного CSS-файла к странице HTML приведен на рисунке 2.4.

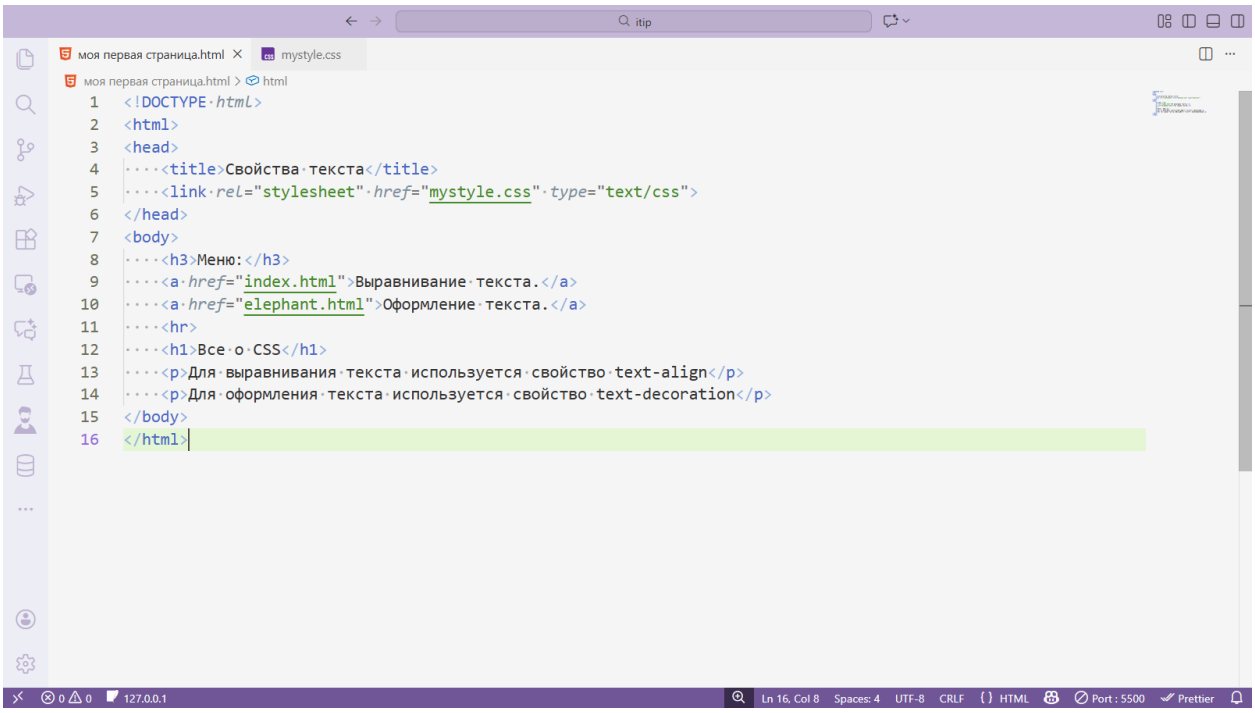



**Рисунок 2.4. Пример подключения внешнего CSS-файла к HTML-документу**

В окне браузера данная страница будет выглядеть как на рисунке 2.5.

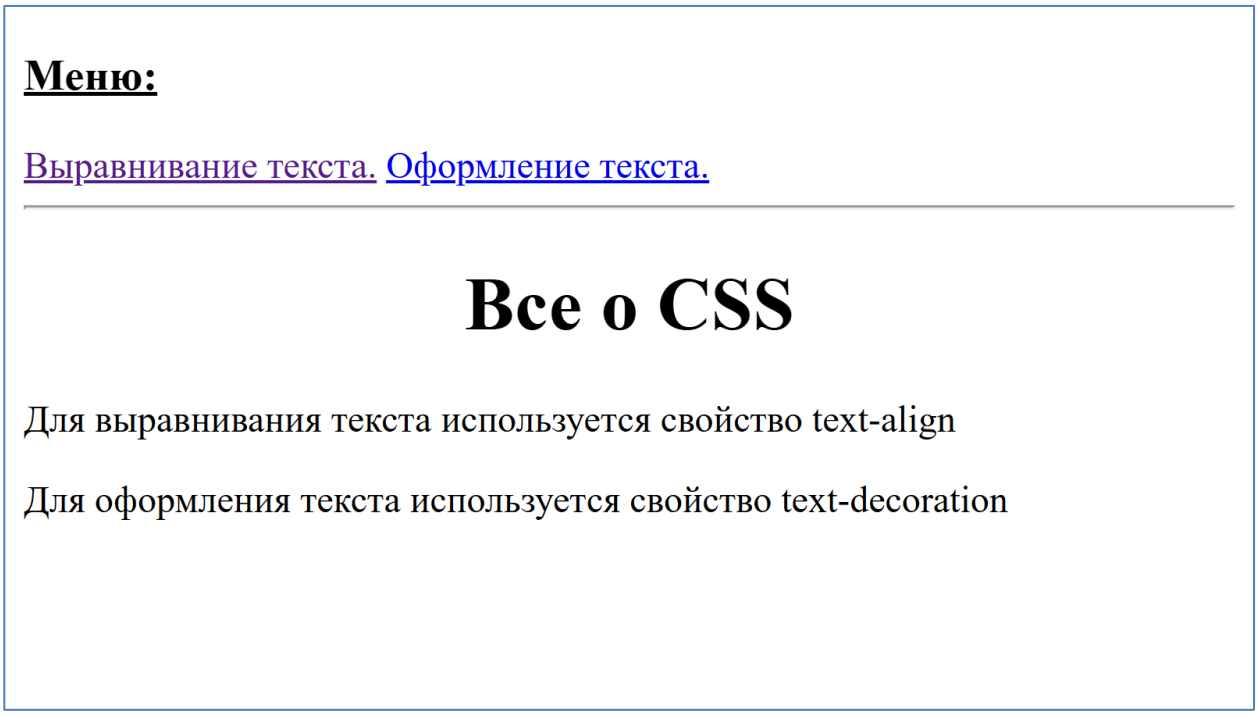



**Рисунок 2.5. HTML-страница в окне браузера**

### Красная строка

Свойство text-indent задаёт отступ первой строки в текстовом блоке с левой стороны.

Размер отступа красной строки может быть задан в процентах от ширины окна браузера или же единицах измерения принятых в CSS.

Пример HTML-кода с использованием text-indent приведен на рисунке2.6.

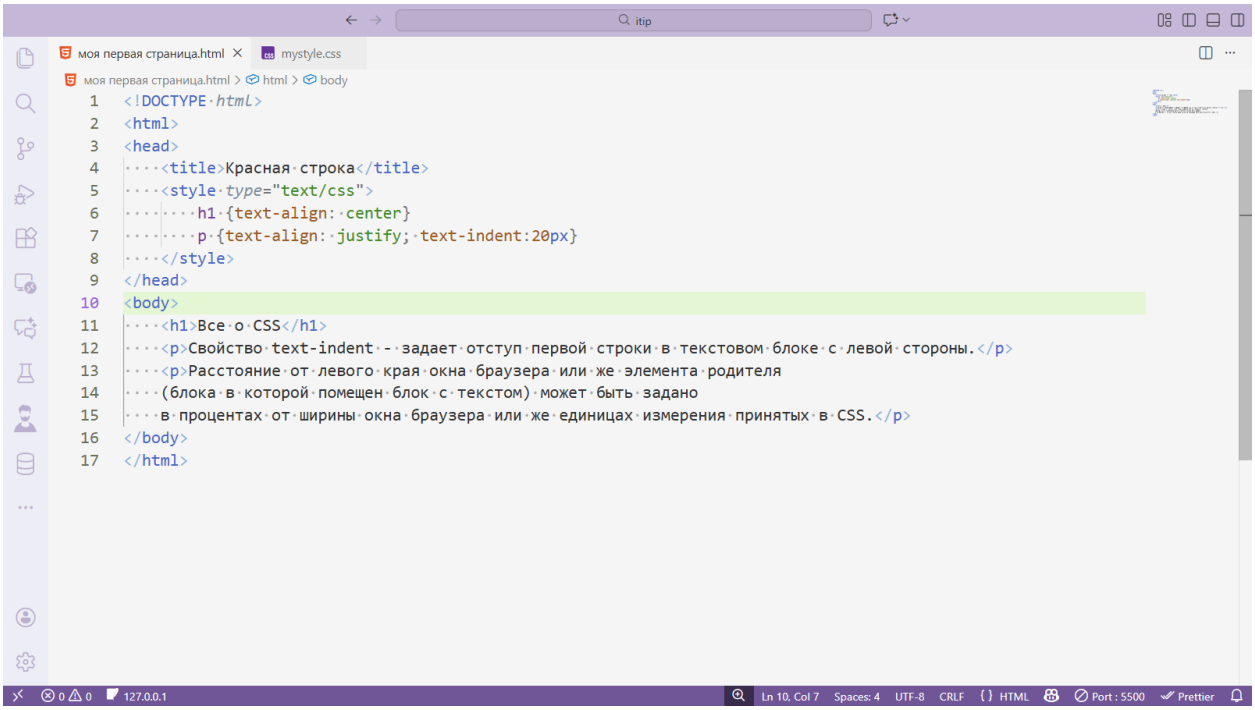



**Рисунок 2.6. Пример использования text-indent в HTML-коде**

В окне браузера данная страница выглядит как на рисунке 2.7.

# Все о CSS

Свойство text-indent - задает отступ первой строки в текстовом блоке с левой стороны.

Расстояние от левого края окна браузера или же элемента родителя (блока в которой помещен блок с текстом) может быть задано в процентах от ширины окна браузера или же единицах измерения принятых в CSS.

**Рисунок 2.7. HTML-страница в окне браузера**

### Трансформация текста

Для трансформации текста используется свойство text-transform. Данное свойство позволяет менять символы в указанном текстовом блоке в соответствии с заданными значениями.

Основные значения данного свойства приведены ниже:

1. None- текст отображается без каких-либо изменений.(по умолчанию)

2. Capitalize - каждое слово в тексте отображается с заглавного символа.

3. Lowercase - все символы преобразуются в нижний регистр.

4. Uppercase - все символы преобразуются в верхний регистр.

Пример использования свойства text-transform в HTML-коде приведен на рисунке 2.8.

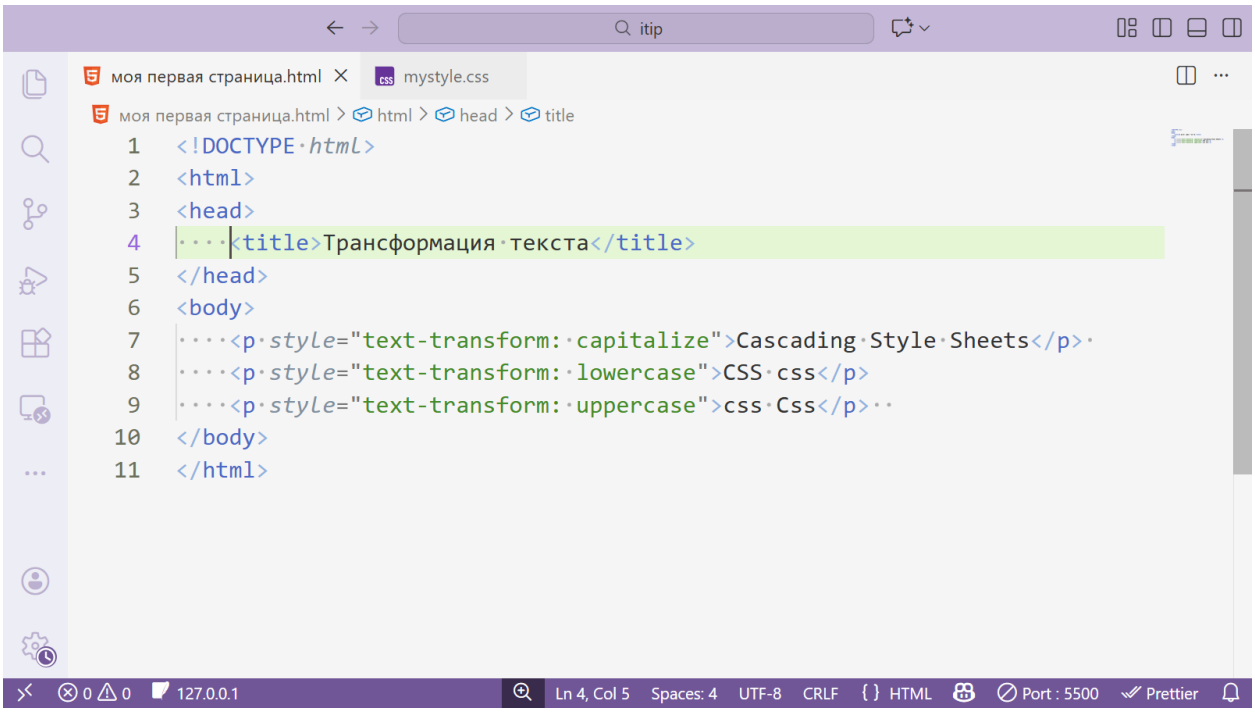



**Рисунок 2.8. Использование свойства text-transform в HTML-коде**

В окне браузера данная страница будет выглядеть как на рисунке 2.9.

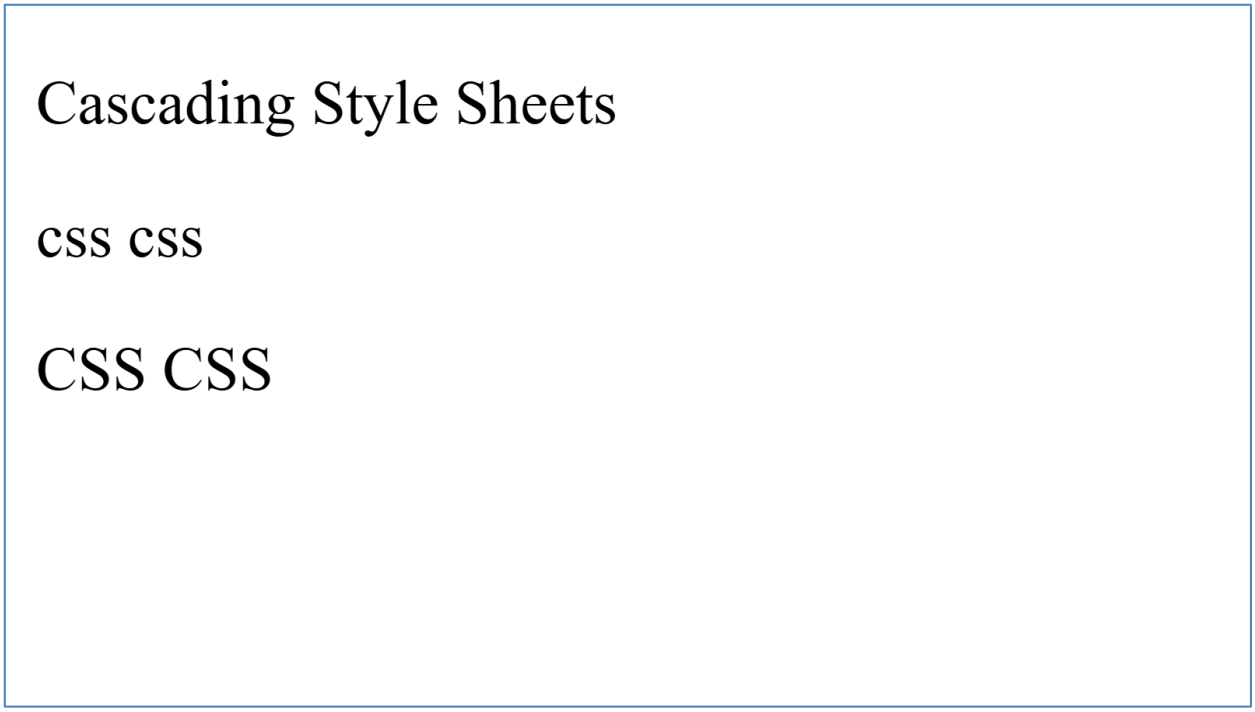



**Рисунок 2.9. HTML-страница в окне браузера**

**Вертикальное выравнивание**

Вертикальное выравнивание текста в строке устанавливает свойство vertical-align

Возможные значения свойства vertical-align:

1. Baseline - выравнивает базовую линию элемента по базовой линии родителя.

2. Bottom - выравнивает элемент по нижней части строки.

3. Middle - выравнивает средину элемента по базовой линии родителя и прибавляет половину высоты родительского элемента.

4. Sub - нижний индекс (размер шрифта не меняется).

5. Super - верхний индекс (размер шрифта не меняется).

6. Text-bottom - нижняя граница элемента выравнивается по нижнему краю строки.

7. Text -top - верхняя граница элемента выравнивается по верхнему краю строки.

8. Top- выравнивает элемент по верхней части строки.

Базовая линия - это линия, на которой располагаются символы в текстовой строке.

Также вертикальное выравнивание элемента относительно строки может выражаться в процентах, пикселях или любых других принятых в CSS единицах измерения. Эти единицы могут принимать как положительные, так и отрицательные значения.

Пример использования свойства vertical-align в HTML-коде приведен на рисунке 2.10.

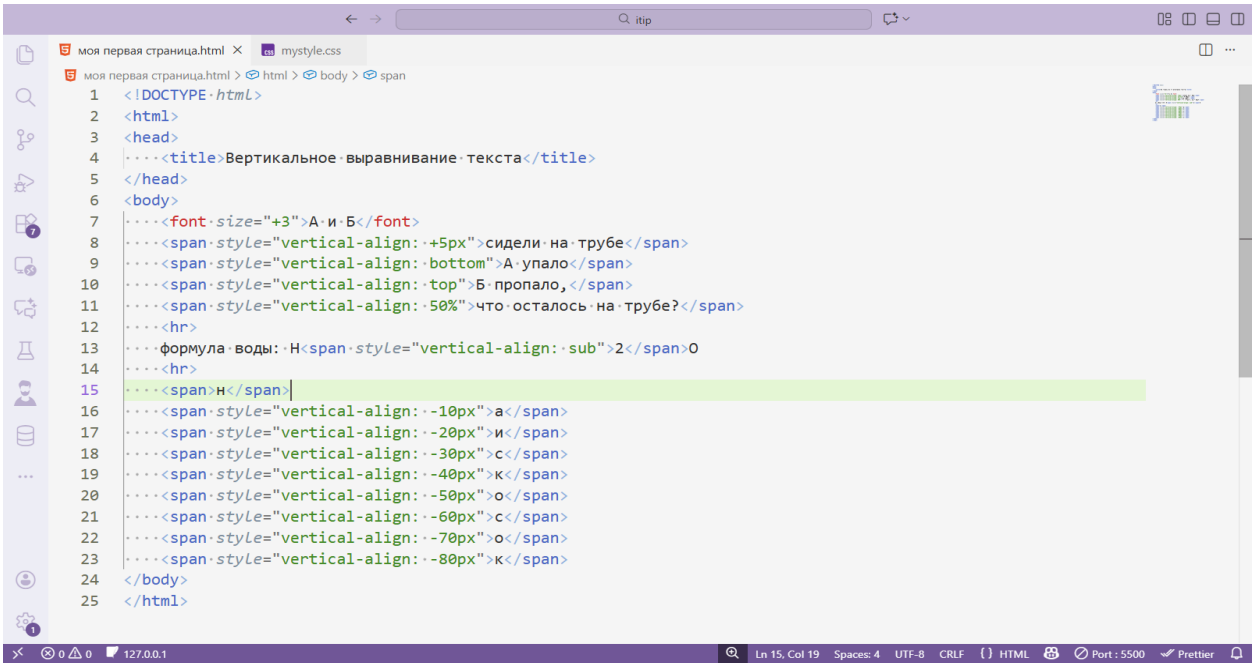



**Рисунок 2.10. Пример использования свойства vertical-align в HTML-коде**

В окне браузера данная страница будет выглядеть как на рисунке 2.11.

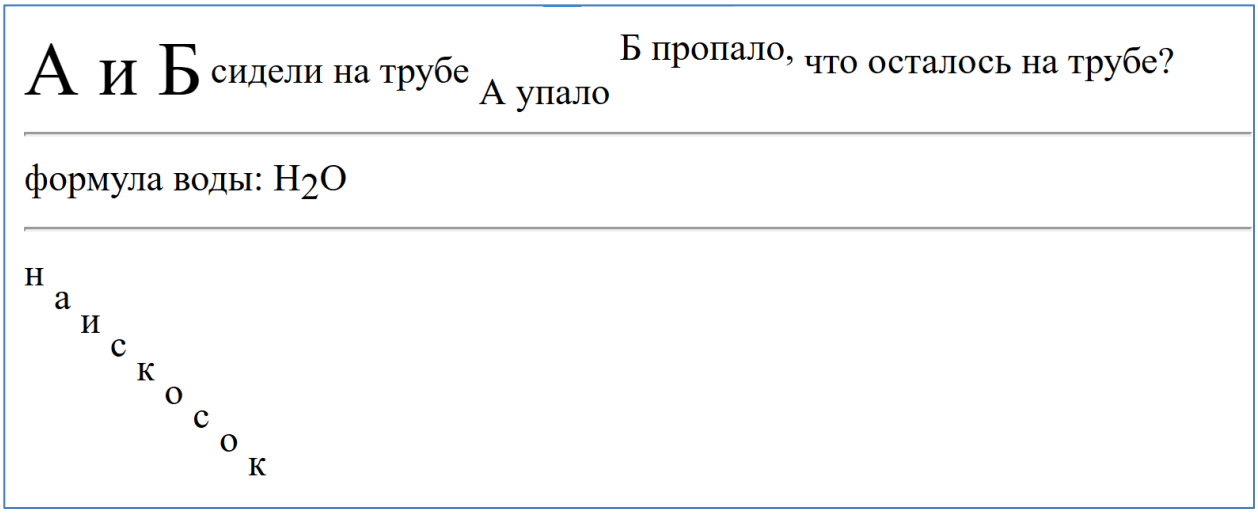



**Рисунок 2.11. HTML-страница в окне браузера**

### Межсимвольное расстояние

Свойство letter-spacing задает расстояние между символами в строке.

Основными значениями данного свойства являются:

1. Normal - нормальное расстояние (по умолчанию).

2. Px - расстояние задаётся в пикселях или любых других единицах измерения принятых в CSS.

Пример использования данного свойства в HTML-коде приведен на рисунке 2.12.

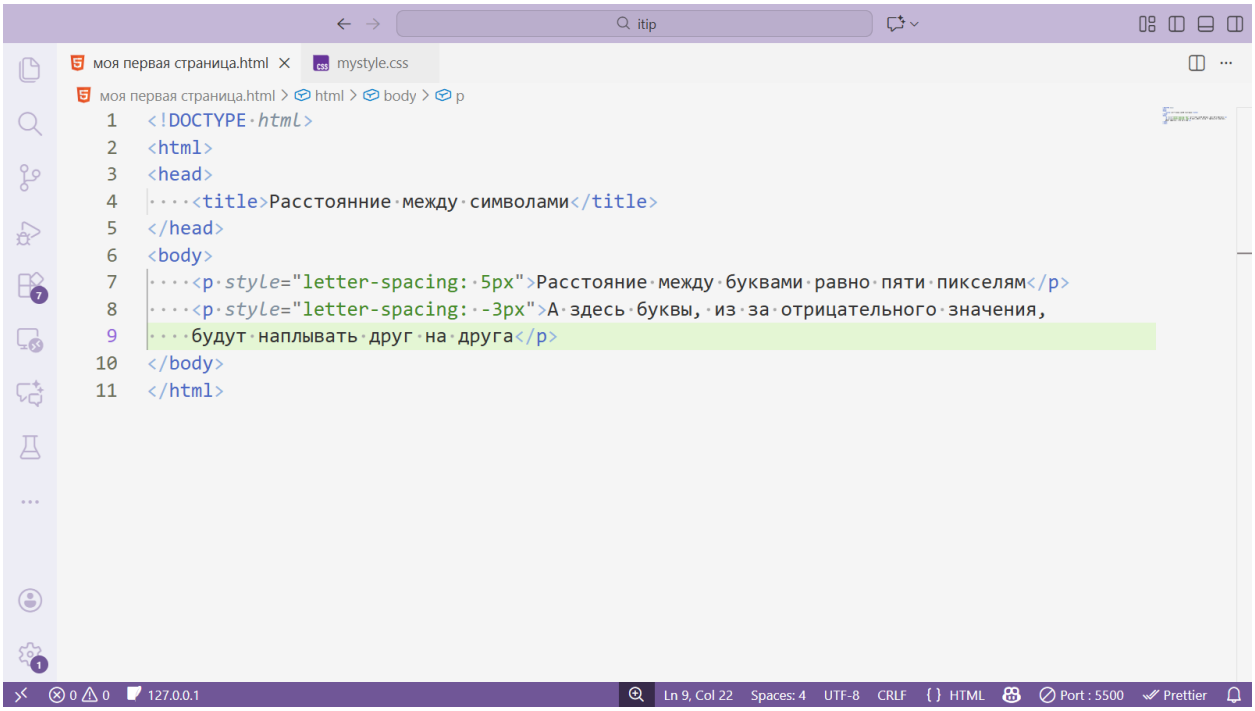



**Рисунок 2.12. Пример использования свойства letter-spacing в HTML-коде**

В окне браузера данная страница будет выглядеть как на рисунке 2.13.

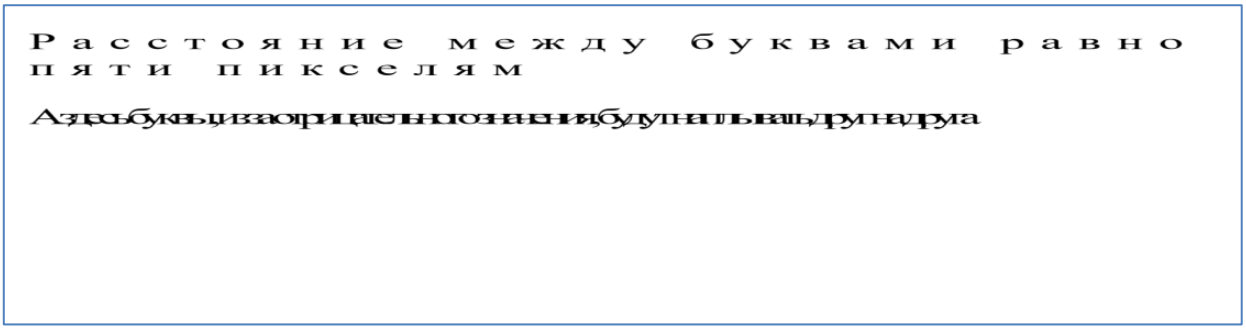



**Рисунок 2.13. HTML-страница в окне браузера**

**Межстрочный интервал.**

Расстояние между строками текста можно задать, используя свойство line-height, сделать это можно следующими способами:

1. Normal - норма (по умолчанию).
2. %- проценты. За 100% берется высота шрифта
3. 0.5- множитель, может быть использовано любое число больше ноля.
4. Px - пиксели и любые другие единицы измерения, принятые в CSS.

Пример использования свойства line-height в HTML-коде приведен на рисунке 2.13.

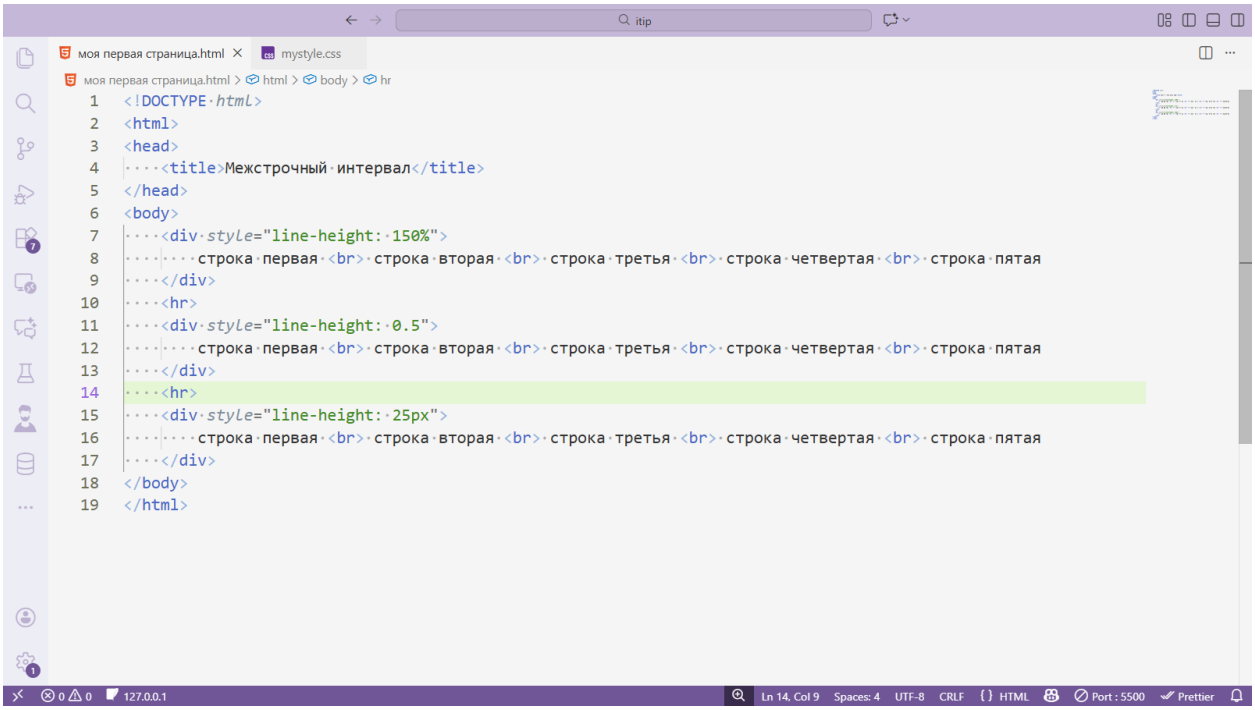



**Рисунок 2.13. Пример использования свойства line-height в HTMLкоде**

В окне браузера данная страница будет выглядеть как на рисунке 2.14.

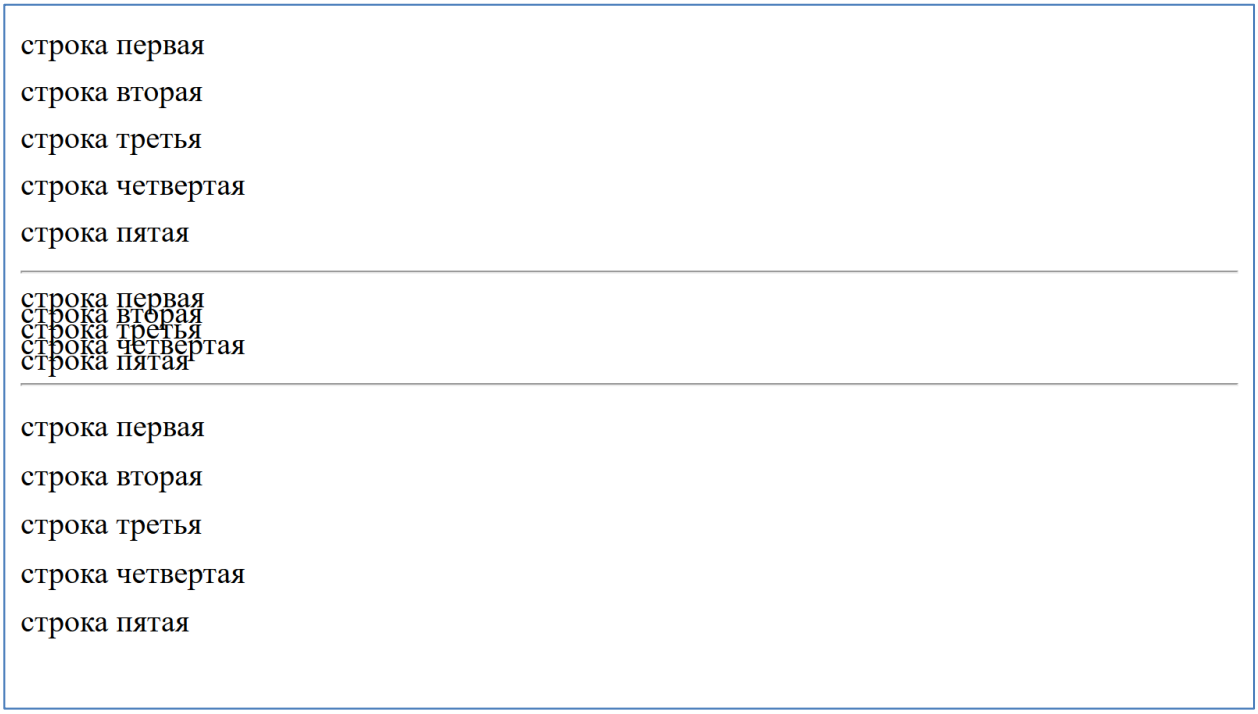



**Рисунок 2.14. HTML-страница в окне браузера**

# РАЗДЕЛ III.
# ВВЕДЕНИЕ В JAVASCRIPT: ОСНОВНЫЕ ПОНЯТИЯ, ОБРАБОТЧИКИ СОБЫТИЙ, ФУНКЦИИ

## Теоретическая справка

Язык программирования JavaScript был разработан фирмой Netscape в сотрудничестве с Sun Microsystems и анонсирован в 1995 году. Изначально *JavaScript* был создан, чтобы *«сделать веб-страницы живыми»*. Основные области использования JavaScript:

- создание динамических страниц, т.е. страниц, содержимое которых может меняться после загрузки;
- проверка правильности заполнения пользовательских форм;
- решение "локальных" задач с помощью сценариев;
- JavaScript-код - основа большинства Ajax-приложений.

### Базовые лексические структуры языка JavaScript.

К базовым лексическим структурам языка JavaScript относят:

- символы;
- чувствительность к регистру;
- пробелы, переводы строк, символы управления форматом;
- точки с запятой;
- комментарии;
- функции.

### Символы

При написании программ на JavaScript используется набор символов Юникода. Юникод является надмножеством кодировок ASCII и Latin-1 и поддерживает практически все письменные языки, имеющиеся на планете. Стандарт ECMAScript 3 требует, чтобы реализации JavaScript обеспечивали поддержку стандарта Юникода версии 2.1 или выше, а стандарт ECMAScript 5 требует, чтобы реализации обеспечивали поддержку стандарта Юникода версии 3 или выше.

### Чувствительность к регистру

JavaScript - это язык, чувствительный к регистру символов. Это значит, что ключевые слова, имена переменных и функций и любые другие идентификаторы языка должны всегда содержать одинаковые наборы прописных и строчных букв.

Например, ключевое слово while должно набираться как «while», а не «While » или «WHILE». Аналогично myvar, Myvar, MyVar и MYVAR - это имена четырех разных переменных. Заметим, однако, что язык разметки HTML (в отличие от XHTML) не чувствителен к регистру. Так как HTML и клиентскийJavaScript тесно связаны, это различие может привести к путанице. Многие JavaScript-объекты и их свойства имеют те же имена, что и теги и атрибуты языка HTML, которые они обозначают. Однако если в HTML эти теги и атрибуты могут набираться в любом регистре, то в JavaScript они обычно должны набираться строчными буквами.

Например, атрибут onclick обработчика события чаще всего задается в HTML как onClick, однако в JavaScript-коде (или в XHTML-документе) он должен быть обозначен как onclick.

### Пробелы, переводы строк, символы управления форматом

JavaScript игнорирует пробелы, которые могут присутствовать между лексемами в программе. Кроме того, JavaScript также по большей части игнорирует символы перевода строки. Поэтому пробелы и символы перевода строки могут без ограничений использоваться в исходных текстах программ для форматирования и придания им удобочитаемого внешнего вида.

Помимо обычного символа пробела (\u0020) JavaScript дополнительно распознает как пробельные следующие символы: табуляция (\u0009), вертикальная табуляция (\u000B), перевод формата (\u000C), неразрывный пробел (\u00A0), маркер порядка следования байтов (\uFEFF), а также все символы Юникода, относящиеся к категории Zs.

Следующие символы распознаются интерпретаторами JavaScript как

символы конца строки: перевод строки (\u000A), возврат каретки (\u000D), разделитель строк (\u2028) и разделитель абзацев (\u2029). Последовательность из символов возврата каретки и перевода строки интерпретируется как единственный символ завершения строки.

Символы Юникода, управляющие форматом (категория Cf), такие как RIGHT-TO-LEFT MARK (\u200F) и LEFT-TO-RIGHT MARK (\u200E), управляют визуальным представлением текста, в котором они присутствуют. Они имеют большое значение для корректного отображения текста на некоторых языках и являются допустимыми в комментариях JavaScript, строковых литералах и в литералах регулярных выражений, но не в идентификаторах (таких как имена переменных), определяемых в программах JavaScript. Исключение составляют ZERO WIDTH JOINER (\u200D) и ZERO WIDTH NON-JOINER (\u200C), которые можно использовать в идентификаторах при условии, что они не являются первыми символами идентификаторов.

### Точки с запятой

Как и в других языках программирования, для отделения инструкций друг от друга в языке JavaScript используется точка с запятой (;). Использование точек с запятой имеет важное значение для ясного выражения намерений программиста: без этого разделителя по ошибке можно принять конец одной инструкции за начало следующей и наоборот.

Обычно в JavaScript точку с запятой между инструкциями можно не ставить, если они находятся в разных строках. (Точку с запятой можно также опустить в конце программы или если следующей лексемой в программе является закрывающая фигурная скобка }.) Многие программисты на JavaScript используют точки с запятой для явного обозначения концов инструкций, даже если в этом нет необходимости.

Взгляните на следующий фрагмент. Поскольку две инструкции находятся в разных строках, первую точку с запятой можно опустить:

```
a = 3
```

```
b = 5;
```

Однако если эти инструкции записать, как показано ниже, первая точка с запятой становится обязательной:

```
a = 3; b = 5;
```

**Комментарии.**

JavaScript поддерживает два способа оформления комментариев. Любой текст между символами // и концом строки рассматривается как комментарий и игнорируется JavaScript. Любой текст между символами /* и */ также рассматривается как комментарий. Эти комментарии могут состоять из нескольких строк, но не могут быть вложенными. Следующие строки представляют собой корректные JavaScript-комментарии:

```
// Это однострочный комментарий.
/* Это тоже комментарий */ // а это другой комментарий.
/*Это еще один комментарий.
 Он располагается в нескольких строках.
*/
```

**Функции.**

Функция - это именованная последовательность операторов (инструкций). Любая функция имеет следующий синтаксис:

```
function имя ()
{
оператор;
.....
оператор;
}
```

Сначала идет ключевое слово function, затем имя функции, затем в круглых скобках перечисляются параметры (если они есть), затем в фигурных скобках перечисляются операторы, т.е. последовательность выполняемых действий. Каждый оператор заканчивается точкой с запятой.

# Лабораторная работа №1

## Размещение кода JavaScript

JavaScript-код можно размещать как на самой странице, так и во внешнем файле.

Пример использования JavaScript-кода на странице приведен на рисунке 1.1.

```
<!DOCTYPE·html>
<html>
<head>
····<title>Заголовок·документа</title>
····<link·rel="stylesheet"·type="text/css"·href="style.css">
····<script·language="javascript"></script>
</head>
<body>
····
</body>
</html>
```

**Рисунок 1. Пример использования JavaScriptна самой странице**

Теоретически теги <*script*></*script*> могут располагаться и в теле документа (в тегах <*body*></*body*>). Выполняться такой сценарий будет тогда, когда браузер "дочитает" страницу до этого места (до тегов <*script*></*script*>). Использовать такой способ сейчас не принято.

Второй способ заключается в использовании внешнего файла, на рисунке 1.2 приведен пример использования этого способа на фрагменте кода сайта Абитуриент ДГУ.

```
<script type="text/javascript">
    (function (d, w, c) {
        (w[c] = w[c] || []).push(function () {
            try {
                w.yaCounter21953365 = new Ya.Metrika({ id: 21953365,
                    webvisor: true,
                    clickmap: true,
                    trackLinks: true,
                    accurateTrackBounce: true
                });
            } catch (e) { }
        });

        var n = d.getElementsByTagName("script")[0],
        s = d.createElement("script"),
        f = function () { n.parentNode.insertBefore(s, n); };
        s.type = "text/javascript";
        s.async = true;
        s.src = (d.location.protocol == "https:" ? "https:" : "http:") + "//mc.yandex.ru/metrika/watch.js";

        if (w.opera == "[object Opera]") {
            d.addEventListener("DOMContentLoaded", f, false);
        } else { f(); }
    })(document, window, "yandex_metrika_callbacks");
</script>
```

**Рисунок 2. Пример использования JavaScript кода во внешнем файле.**

## Первая программа

Создадим первую программу на Html с использованием внешнего файла .js.

Для этого откройте любой текстовый редактор (желательно Notepad++) и впишите код, как на рисунке 1.3.

```
function showMessage() {
  alert('Вы щелкнули по div-y');
}
```

**Рисунок 1.3. Фрагмент первой программы в консоли браузера**

Затем сохраняем данный файл в папке с лабораторной работой под названием script.js (рисунок 1.4).

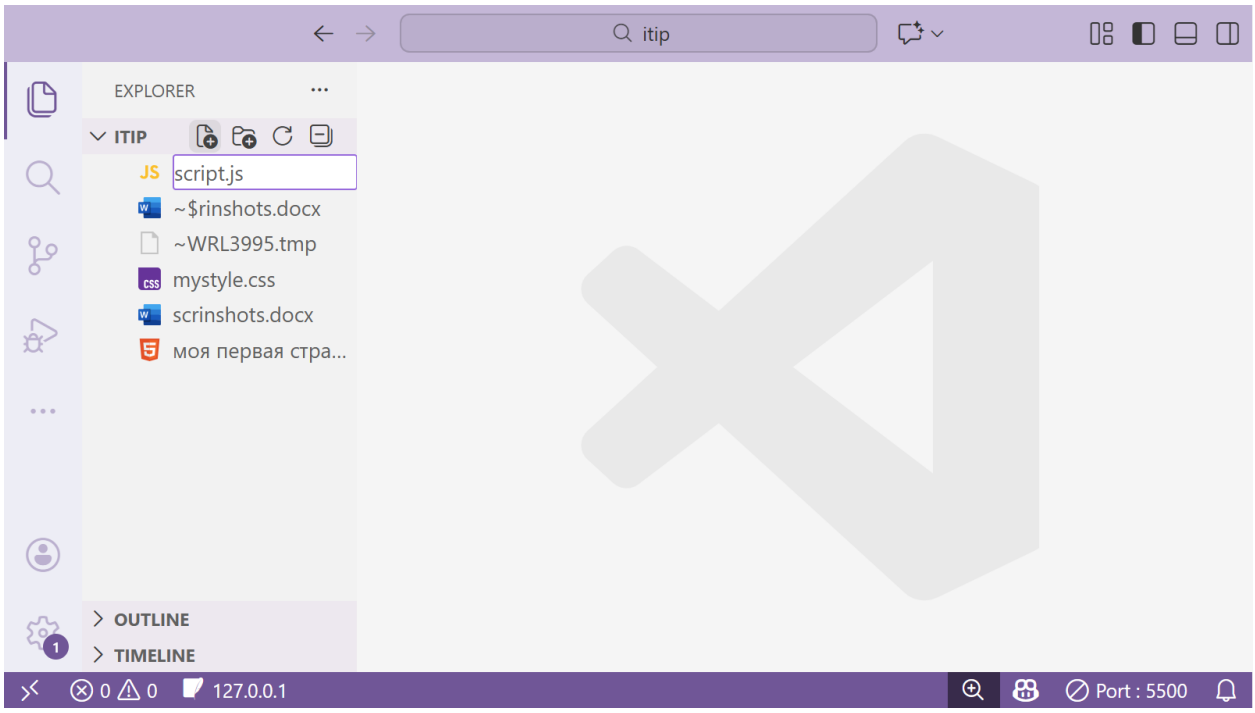



**Рисунок 1. 4. JavaScript-код во внешнем файле**

**!Примечание:** перед сохранением файла проверьте кодировку - необходимо установить UTF-8.

Далее создадим html файлом со ссылкой на наш файл script.js (рисунок 1.5).

```
<html>
<head>
    <title>Заголовок документа</title>
    <link rel="stylesheet" type="text/css" href="style.css">
    <script type="text/javascript" src="script.js"></script>
</head>
<body>
    <div onClick="showMessage();">Щелкни меня</div>
</body>
</html>
```

**Рисунок 1.5. Пример html и javascriptкодов в окне браузера**

**!Примечание:** если введенный вами в файл русский текст отображается в окне браузера нечитаемыми символами («кракозябры»), проверьте кодировку (должно быть UTF-8), если же с кодировкой все нормально, а проблема так и не решилась, то попробуйте вписать код <metacharset="UTF-8" />

между тегами <head>.

Как видно из рисунка в html-коде мы использовали обработчик событияonClick на строку«Щелкни меня», которая служит названием функции (перечень наиболее часто используемых событий с их описанием приведен в таблице 1.1 в конце лабораторной работы). В самом же файле script.js мы описываем уже функцию, а точнее те действия, которые она выполняет при обращении к ней.

В окне браузера выполненный код будет выглядеть как на рисунке 1.6.

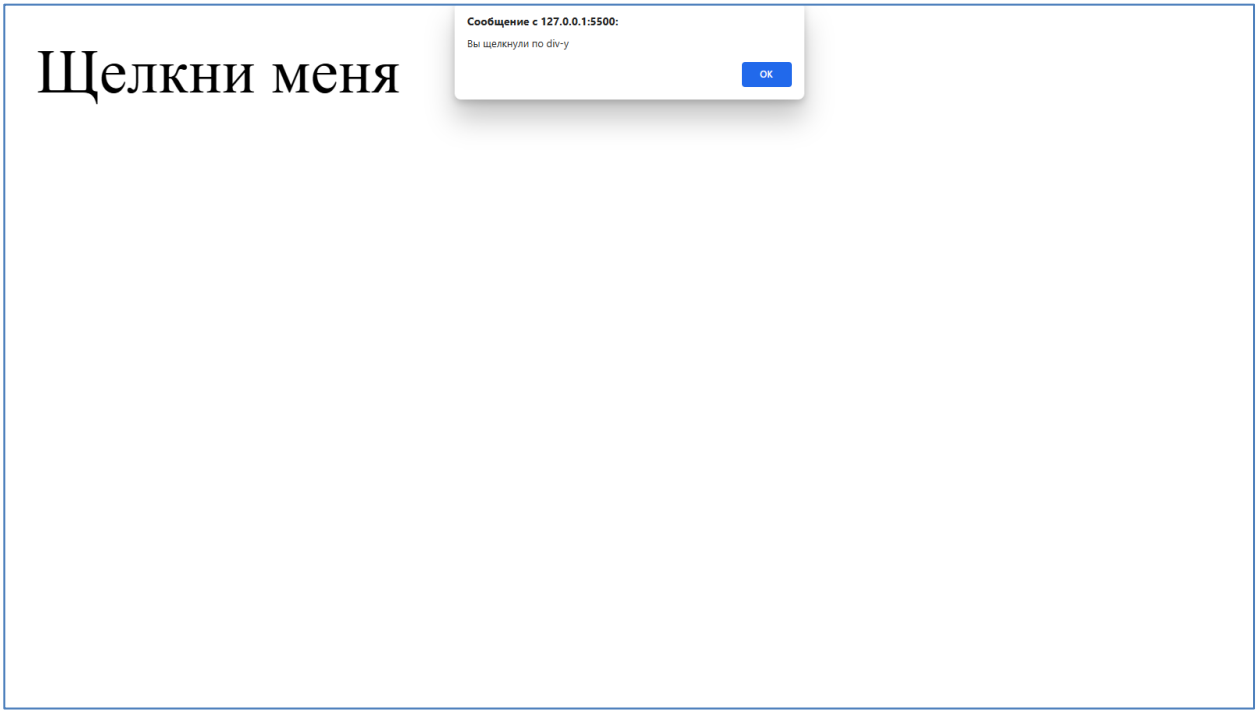



**Рисунок 1.6. Html-страница в окне браузера**

Выполним теперь программу чуть сложнее.

Напишем программу для расчета площади прямоугольника.

Для этого снова откроем окно редактора Visual Studio Code и впишем код для html-файла (рисунок 1.7).

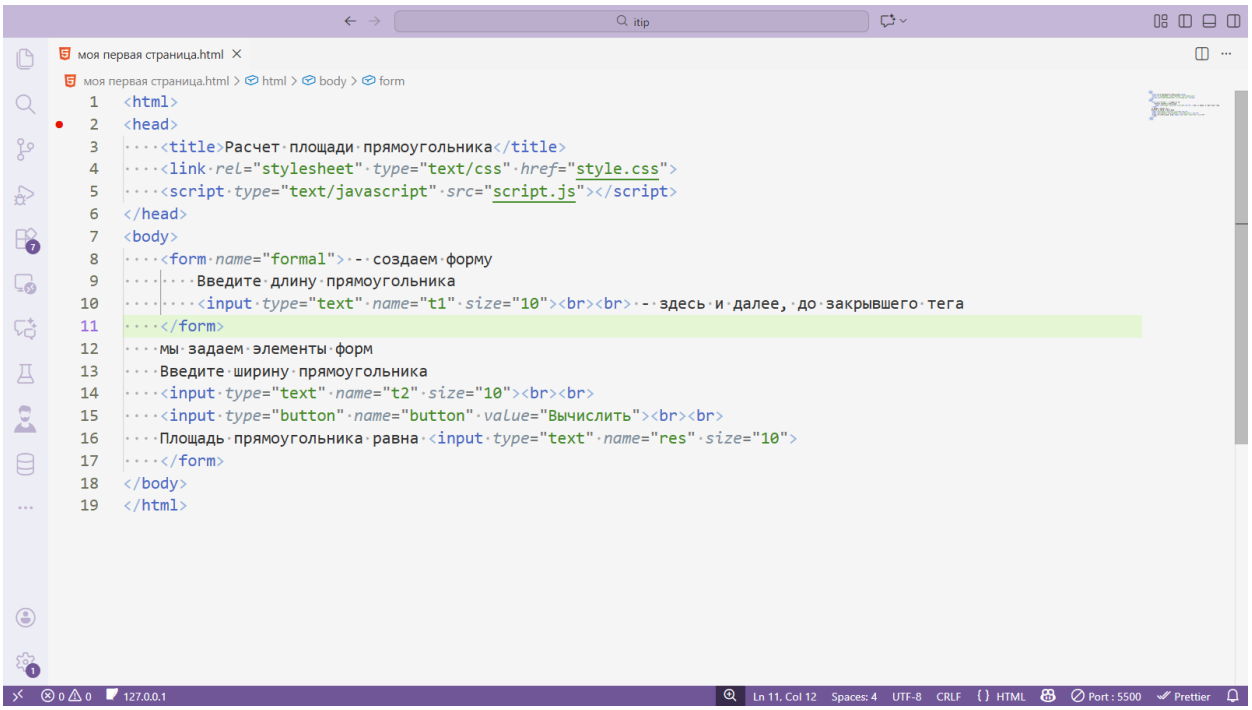



**Рисунок 1.7. Код программы для расчета площади прямоугольника**

В окне браузера эта страница будет выглядеть как на рисунке 1.8.

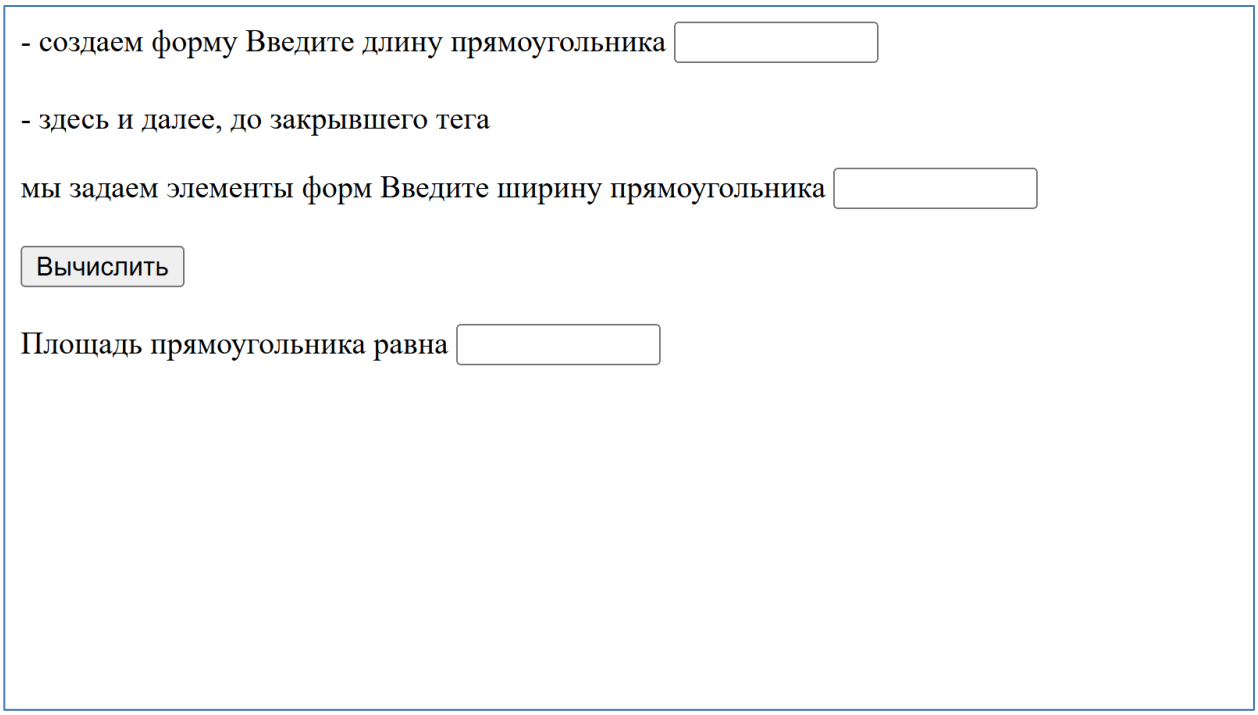



**Рисунок 1.8.Html-страница примера расчета площади прямоугольника в окне браузера**

Как видно из рисунка мы задали визуальные элементы для ввода исходных данных и вывода результата вычисления, а также кнопку для обработчика событий.

Следующим шагом мы определяем функцию, которая вызывается с помощью обработчика событий. Для этого к элементу Button мы добавляем следующий фрагмент кода:

**onClick="areaRectangle (рисунок 1.9)**

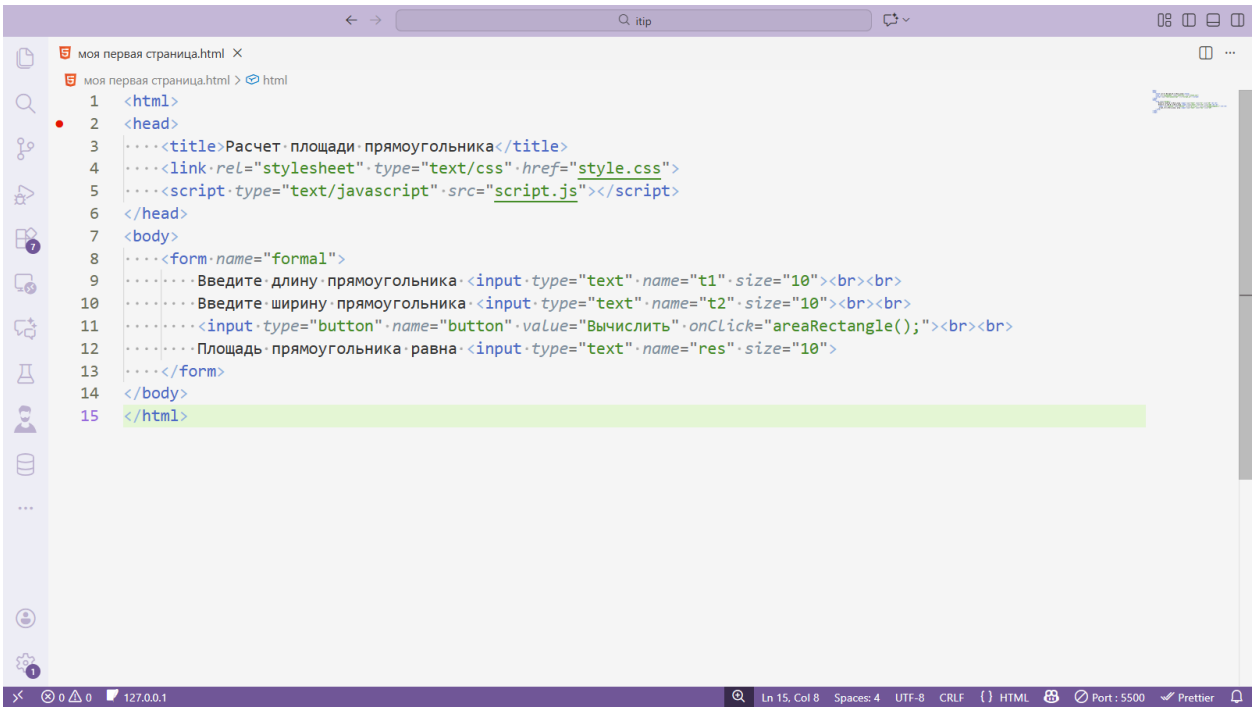



**Рисунок 1.9. Код программы для расчета площади прямоугольника с помощью обработчика события onClick**

Таким образом, у нас теперь есть функция areaRectangle, тело которой мы пропишем в файле script.js.

Вспомним для начала, что площадь прямоугольника находится путем умножения длины прямоугольника на ширину. Обозначим их через параметры a и b, соответственно. Теперь нам необходимо дать нашей функции «понять», откуда брать значения наших переменных для выполнения расчетов.

Вернемся к форме, созданной нами ранее (рисунок 1.10).

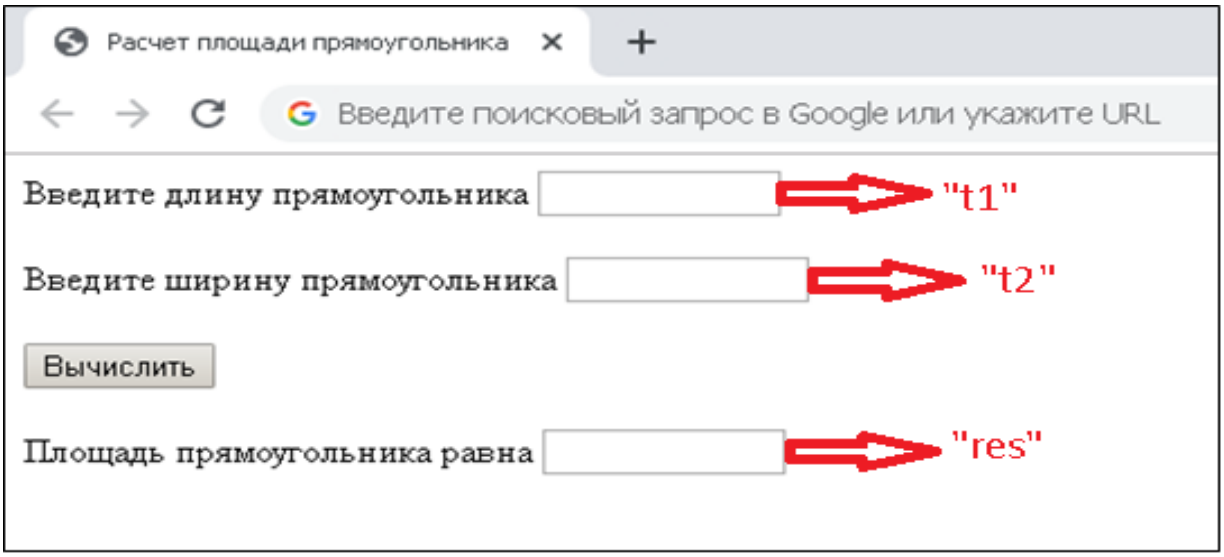



**Рисунок 1.10. Htmlстраница примера расчета площади прямоугольника в окне браузера**

В коде, который мы прописали ранее в файле html, мы задали названия элементов форм. Так, входные данные пользователь будет вписывать в элементы формы с названиями t1 и t2, а значение рассчитываемого показателя будет отображаться в элементе res.

Таким образом, мы должны в файле script.js, при описании функции объявить переменные и указать, из каких элементов форм мы берем их значения:

var - блок объявления переменных;

a - название первой переменной;

document - даем понять, что значения показателей нужно брать из текущей страницы;

forma1 - задаем название формы, элементы которой содержат исходные и расчетные данные;

t1 - объект формы, содержащий значение первой переменной;

value - значение переменной из элементы t1.

Соединив все составные элементы, получим следующую строчку кода:

**vara=document.forma1.t1.value;**

Далее нам необходимо прописать саму формулу расчета и указать, в какой элемент формы необходимо вписать полученное значение:

**s=a*b**-формула расчета площади;

**document.forma1.res.value=s** -вписываем расчетное значение в элемент формы res;

Соединив все строчки, получим код, как на рисунке 1.11.

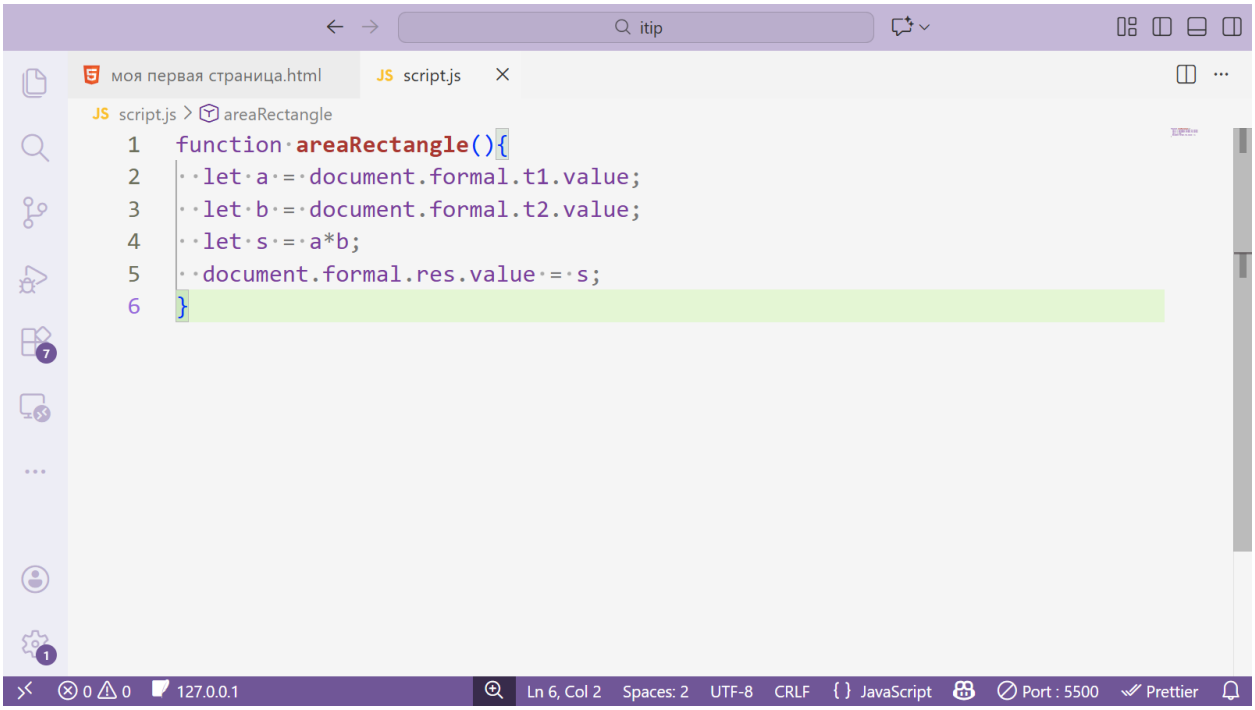



**Рисунок 1.11. Фрагмент программы в консоли браузера**

Результат работы, прописанного нами кода приведен на рисунке 1.12.

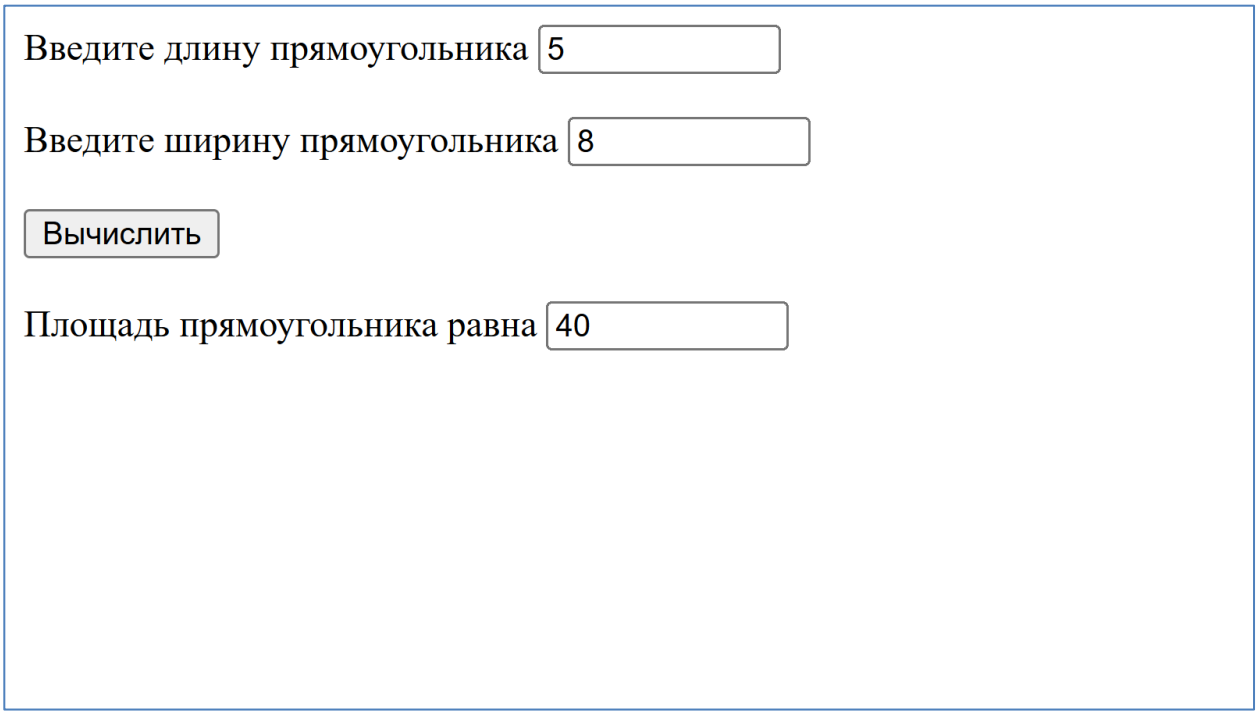



**Рисунок 1.12. Пример расчета площади круга в окне браузера**

## Справочные материалы

*Таблица 1*

*События, поддерживаемые JavaScript*

| Событие | Когда происходит | Обработчик события |
|---|---|---|
| Blur | потеря объектом фокуса | onBlur |
| Change | пользователь изменяет значение элемента | onChange |
| Click | пользователь щелкает мышью по объекту | onClick |
| DblClick | пользователь делает двойной щелчок мышью по объекту | onDblClick |
| DragDrop | пользователь перетаскивает мышью объект | onDragDrop |
| Error | возникновение javascript-ошибки | onError |
| Focus | окно или элемент формы получает фокус | onFocus |
| KeyDown | пользователь нажимает клавишу клавиатуры | onKeyDown |
| KeyPress | пользователь удерживает нажатой клавишу клавиатуры | onKeyPress |
| KeyUp | пользователь отпускает клавишу клавиатуры | onKeyUp |
| Load | документ загружается в браузер | onLoad |
| MouseDown | пользователь нажимает кнопку мыши | onMouseDown |
| MouseOut | указатель мыши выходит за пределы элемента | onMouseOut |
| MouseOver | указатель мыши помещается над элементом | onMouseOver |
| MouseUp | пользователь отпускает кнопку мыши | onMouseUp |
| Move | пользователь перемещает окно | onMove |
| Reset | пользователь нажимает кнопку "reset" формы | onReset |
| Resize | пользователь изменяет размеры окна или элемента | onResize |
| Select | пользователь выбирает элемент формы | onSelect |
| Submit | пользователь нажимает кнопку "submit" формы | onSubmit |
| Unload | пользователь закрывает документ | onUnload |

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ


**1.** *Баммаева Г. А. Интернет программирование учебно-методическое пособие / Г. А. Баммаева, Москва : [б. и.];, Перо, 2020. - 96 с. ил., табл.; 21 см. - ISBN 978-5-00171-565-8.*

**2.** *Беликова С.А. Основы HTML и CSS: проектирование и дизайн веб-сайтов : учебное пособие по курсу «Web-разработка» / Беликова С.А., Беликов А.Н.. - Ростов-на-Дону, Таганрог : Издательство Южного федерального университета, 2020. - 174 с. - ISBN 978-5-9275-3435-7.*

**3.** *Минязев, Р. Ш. Скриптовые языки web-программирования (JavaScript, PHP, html/CSS): учебно-методическое пособие / Р. Ш. Минязев. - Казань : КНИТУ-КАИ, 2022. - 60 с. - ISBN 978-5-7579-2632-2.*

**4.** *Никулова, Г. А. Web-технологии: Введение в программирование на JavaScript: защита контента средствами JS и CSS : учебно-методическое пособие / Г. А. Никулова, А. С. Терлецкий. - Липецк : Липецкий ГПУ, 2023. - 76 с. - ISBN 978-5-907792-00-5.*

**5.** *Петракова, Н. В. Основы CSS: учебно-методическое пособие / Н. В. Петракова. - Брянск : Брянский ГАУ, 2023 - Часть 2 - 2023. - 107 с.*

**6.** *Петракова, Н. В. Основы HTML: учебно-методическое пособие / Н. В. Петракова. - Брянск : Брянский ГАУ, 2022 - Часть 1 - 2022. - 50 с.*

**7.** *Побединский, Е. В. Проектирование веб-сайтов с использованием технологий PHP, HTML, CSS и WordPress : учебное пособие / Е. В. Побединский, В. В. Побединский. - Екатеринбург : УГЛТУ, 2018. - 115 с. - ISBN 978-5-94984-651-3.*

**8.** *Фролов А.Б. Web-сайт. Разработка, создание, сопровождение: учебное пособие / Фролов А.Б., Нагаева И.А., Кузнецов И.А.. - Саратов : Вузовское образование, 2024. - 355 c. - ISBN 978-5-4487-1025-4.*

**9.** *Фрэйн Б. HTML5 и CSS3. Разработка сайтов для любых браузеров и устройств. 2-е изд. - СПб.: Питер, 2017. - 272 с.: ил. - (Серия «Библиотека программиста»).ISBN 978-5-496-02271-2*

**10.** *Янцев, В. В. Разработка web-страниц на HTML, CSS и JavaScript : учебное пособие для вузов / В. В. Янцев. - Санкт-Петербург : Лань, 2024. - 148 с. - ISBN 978-5-507-49640-2.*




Баммаева Г.А., Мусаева У.А.

# Интернет программирование:

*Учебное методическое пособие*

Издательство «Перо»
109052, Москва, Нижегородская ул., д. 29-33, стр. 15, ком. 536
Тел.: (495) 973-72-28, 665-34-36
Подписано в печать 22.12.2025. Формат 60×90/16.
Бумага офсетная. Усл. печ. л. 5,625. Тираж 600 экз. Заказ 1386.