

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Оренбургский государственный университет»

Л.Ф. Тагирова

РАЗРАБОТКА КЛИЕНТСКИХ ВЕБ-ПРИЛОЖЕНИЙ

Учебное пособие

Рекомендовано ученым советом федерального государственного образовательного учреждения высшего образования «Оренбургский государственный университет» для обучающихся по образовательным программам высшего образования по направлению подготовки 09.03.04 Программная инженерия

Оренбург
2025

УДК 004.41(075.8)
ББК 32.973.4я73
Т13

Рецензент - доктор технических наук, профессор Н.А. Соловьев

Т13 **Тагирова, Л. Ф.**
Разработка клиентских веб-приложений: учебное пособие /
Л.Ф. Тагирова; Оренбургский гос. ун-т. - Оренбург: ОГУ, 2025. - 305с.

ISBN

Учебное пособие посвящено изучению создания клиентских веб-приложений с использованием языка разметки HTML5 и языка стилей CSS3.

Данное учебное пособие может быть использовано обучающимися направления подготовки 09.03.04 Программная инженерия, профиля “Разработка программно-информационных систем” при изучении дисциплины “Программирование web-приложений”, направления подготовки 09.03.04 Программная инженерия, профиля “Веб- и мобильные приложения” при изучении дисциплины “Разработка динамических сайтов”.

УДК 004.41(075.8)
ББК 32.973.4я73

© Тагирова Л.Ф., 2025
© ОГУ, 2025

Содержание

Введение	6
1 Введение в HTML.....	8
1.1 История развития языка HTML	8
1.2 Элементы и атрибуты HTML5	9
1.3 Создание документа HTML5.....	12
1.4 Элемент head и метаданные веб-страницы.....	14
1.5 Элементы группировки в HTML5	17
1.6 Заголовки в HTML5.....	20
1.7 Форматирование текста	21
1.8 Escape-последовательности в HTML	23
1.9 Работа с изображениями.....	25
1.10 Мультимедиа.....	27
1.11 Списки в HTML	31
1.12 Элемент details	35
1.13 Обозначение цветов в HTML	36
1.14 Таблицы в HTML5	40
1.15 Гипертекстовые ссылки в HTML	42
1.16 Элементы figure и figcaption.....	47
1.17 Использование плавающих фреймов в HTML	48
1.18 Контрольные вопросы.....	50
1.19 Тестирование по главе	50
2 Работа с формами в HTML5	53
2.1 Задание формы.....	53
2.2 Элементы форм.....	55
2.3 Кнопки	58
2.4 Текстовые поля	60
2.5 Поле ввода пароля	64
2.6 Метки и автофокус	64
2.7 Элементы для ввода чисел.....	66
2.8 Флажки и переключатели	69
2.9 Элементы для ввода цвета, url, email, телефона.....	71
2.10 Элементы для ввода даты и времени.....	74
2.11 Отправка файлов.....	76
2.12 Список select	77
2.13 Textarea	80
2.14 Валидация форм	81
2.15 Контрольные вопросы.....	89
2.16 Тестирование по главе	90
3 Семантическая верстка страниц в HTML5	93
3.1 Элемент article.....	94
3.2 Элементы section	95
3.3 Элемент nav.....	97

3.4 Элемент header.....	99
3.5 Элемент footer.....	99
3.6 Элемент address	101
3.7 Элемент aside	102
3.8 Элемент main.....	103
3.9 Контрольные вопросы.....	104
3.10 Тестирование по главе	104
4 Работа с каскадными таблицами стилей	107
4.1 Виды каскадных таблиц стилей	108
4.2 Задание стилей с помощью селекторов	110
4.3 Селекторы потомков.....	117
4.4 Селекторы дочерних элементов.....	120
4.5 Селекторы элементов одного уровня	122
4.6 Псевдоклассы.....	124
4.7 Псевдоклассы дочерних элементов	126
4.8 Псевдоклассы форм.....	130
4.9 Селекторы атрибутов	135
4.10 Наследование и каскадность стилей.....	137
4.11 Контрольные вопросы.....	141
4.12 Тестирование по главе	142
5 Фильтры в CSS.....	145
5.1 Назначение фильтров	145
5.2 Контрольные вопросы.....	150
5.3 Тестирование по главе.....	150
6 Блочные элементы в CSS	153
6.1 Структура блочного элемента.....	153
6.2 Внешние отступы	154
6.3 Внутренние отступы	155
6.4 Границы	157
6.5 Фон блочного элемента	161
6.6 Позиционирование блочных элементов	170
6.7 Создание панели навигации	193
6.8 Использование свойства box-sizing	197
6.9 Создание макета веб-страницы из блочных элементов	199
6.10 Контрольные вопросы.....	204
6.11 Тестирование по главе.....	204
7 Трансформации, переходы и анимации.....	207
7.1 Трансформации.....	207
7.2 Переходы	215
7.3 Анимация в CSS3	220
7.4 Контрольные вопросы.....	228
7.5 Тестирование по главе.....	228
8 Адаптивная верстка.....	231
8.1 Введение в адаптивный дизайн.....	231

8.2 Media Query в CSS	231
8.3 Контрольные вопросы.....	238
8.4 Тестирование по главе.....	238
9 Создание гибкого макета страницы с помощью Flexbox	241
9.1 Понятие Flexbox. Контейнер Flex.....	241
9.2 Направление flex-direction	244
9.3 Flex-flow. Порядок элементов	248
9.4 Выравнивание элементов. Justify-content.....	250
9.5 Выравнивание элементов. Свойство align-items	254
9.6 Многоколоночный дизайн на Flexbox	256
9.7 Пример макета страницы на Flexbox.....	260
9.8 Контрольные вопросы.....	263
9.9 Тестирование по главе.....	263
10 Двумерная система сеток Grid Layout	266
10.1 Определение Grid Layout. Grid Container.....	266
10.2 Строки и столбцы	268
10.3 Функция repeat и свойство grid.....	273
10.4 Размеры строк и столбцов	274
10.5 Отступы между столбцами и строками.....	278
10.6 Позиционирование элементов в системе сеток Grid Layout.....	279
10.7 Направление и порядок элементов	283
10.8 Создание макета страницы в Grid Layout	287
10.9 Контрольные вопросы.....	290
10.10 Тестирование по главе	290
11 Использование переменных в CSS	293
11.1 Стилизация с помощью переменных.....	293
11.2 Создание тем CSS с помощью переменных	295
11.3 Стили CSS как хранилище данных.....	298
11.4 Контрольные вопросы.....	300
11.5 Тестирование по главе.....	300
Заключение.....	303
Список использованных источников.....	304

Введение

Разработка современных веб-приложений требует использования современных средств и технологий для создания клиентской части приложений. На сегодняшний день для работы на стороне клиента используется новая версия языка разметки – HTML5, с применением стилизации страниц с помощью CSS3.

Язык HTML - стандартизированный язык гипертекстовой разметки документов для просмотра веб-страниц в браузере. Элементы HTML являются строительными блоками веб-страниц. Язык разметки HTML предоставляет средства для форматирования различных элементов, таких как заголовки, абзацы и списки, гиперссылки. Включение использования в HTML-документы каскадных таблиц CSS позволяет задавать стилизованный внешний вид страницы.

CSS представляет собой формальный язык декорирования и описания внешнего вида документа, написанного с использованием языка разметки. Разработчики используют CSS для определения внешнего вида веб-страниц, включая цвета, шрифты и стили, а также размещение блоков. Основной целью внедрения элементов CSS является отделение описания логической структуры веб-страницы от описания ее внешнего вида.

Целью данного пособия является обучение студентов современным средствам и подходам к разработке клиентской части сайтов, приобретение навыков разработки и отладки веб-приложений с использованием языка разметки HTML5 и каскадных таблиц стилей CSS3.

Пособие состоит из двух одиннадцати глав, касающихся изучения языка HTML5 (1-3 глава) и каскадных таблиц стилей CSS3 (4-10 глава) соответственно. В первой главе представлен материал по изучению основ языка разметки HTML5. Рассматриваются такие элементы форматирования как заголовки, работа с текстом, escape-последовательности, изображения, мультимедиа. Также в первой главе описана работа со списками, цветами, таблицами и плавающими фреймами.

Во второй главе рассматриваются вопросы, касающиеся работы с формами в HTML5. Представлены примеры использования таких элементов как однострочные и многострочные текстовые поля, кнопки, флажки и переключатели, списки и др.

Третья глава предоставляет информацию по семантической структуре страницы. Изучаются теги `article`, `section`, `nav`, `header`, `aside`, `main`. Четвертая глава посвящена изучению основ работы с каскадными таблицами стилей CSS3. Рассматриваются внутренние, внешние и внедренные таблицы стилей, а также различные виды селекторов. В пятой главе рассматриваются статические фильтры CSS3 (`blur`, `brightness`, `contrast`, `grayscale`, `hue-rotate`, `invert`, `opacity`, `saturate`, `sepia`).

Шестая глава посвящена изучению блоковых элементов в CSS2, заданию границ, отступов, а также позиционированию. Седьмая глава посвящена изучению трансформаций, переходов и анимаций. В восьмой главе изучается адаптация содержимого сайта для различных типов экранов мониторов (компьютер, планшет, мобильный телефон), на которых запущено веб-приложение.

В девятой главе описано создание гибкого макета страницы с помощью Flexbox. Рассматривается контейнер `flex`, задание направления и порядок элементов в контейнере, выравнивание элементов. Десятая глава посвящена проектированию двумерной системы сеток Grid Layout. Описывается задание строк и столбцов, направления и порядка элементов, создания макета страницы в Grid Layout.

В одиннадцатой главе рассматривается применение переменных для задания свойств в CSS. Описано создание стилей, тем CSS с помощью переменных, а также применение CSS как хранилище данных.

Для более наглядного представления теоретический материал учебного пособия дополняется практическими примерами и наглядными изображениями. После окончания изучения представленного учебного пособия студенты будут уметь самостоятельно разрабатывать веб-приложения, работающие на стороне клиента с использованием возможностей HTML5 и CSS3.

Автор выражает глубокую благодарность и признательность рецензенту за внимательное прочтение рукописи и замечания, способствовавшие улучшению качества предлагаемого учебного пособия.

1 Введение в HTML

1.1 История развития языка HTML

HTML (HyperText Markup Language) представляет язык разметки гипертекста, используемый преимущественно для создания документов в сети интернет. HTML начал свой путь в начале 90-х годов как примитивный язык для создания веб-страниц. В 2014 году официально была завершена работа над новым стандартом - HTML5, который произвел революцию, привнеся в HTML много нового:

- 1 HTML5 определяет новый алгоритм автоматизированного получения информации с веб-сайтов для создания структуры DOM;

- 2 добавление новых элементов и тегов, как например, элементы video, audio и ряд других;

- 3 переопределение правил и семантики уже существовавших элементов HTML.

Фактически с добавлением новых функций HTML5 стал не просто новой версией языка разметки для создания веб-страниц, но и фактически платформой для создания приложений, а область его использования вышла далеко за пределы веб-среды интернет: HTML5 применяется также для создания мобильных приложений под Android, iOS, Windows Mobile и даже для создания десктопных приложений для обычных компьютеров [2].

Развитием HTML5 занимается независимая международная организация под названием World Wide Web Consortium (сокращенно W3C, Консорциум Всемирной паутины), которая определяет стандарт HTML5 в виде спецификации. W3C продолжает работать над развитием HTML5 и публиковать обновления.

Большинство последних версий браузеров поддерживают большинство функциональности HTML5 (Yandex, Firefox, Opera, Internet Explorer 11, Microsoft

Edge). В то же время многие старые браузеры, как например, Internet Explorer 8 и более младшие версии, не поддерживают стандарты, а IE 9, 10 поддерживает лишь частично. При этом даже те браузеры, которые в целом поддерживают стандарты, могут не поддерживать какие-то отдельные функции.

1.2 Элементы и атрибуты HTML5

Прежде чем переходить непосредственно к созданию своих веб-страниц на HTML5, рассматриваются основные строительные блоки, кирпичики, из которых состоит веб-страница.

Документ HTML5, как и любой документ HTML, состоит из элементов, а элементы состоят из тегов. Как правило, элементы имеют открывающий и закрывающий тег, которые заключаются в угловые скобки. Например:

<div>Текст в элементе div</div>

Здесь определен элемент div, который имеет открывающий тег <div> и закрывающий тег </div>. Между этими тегами находится содержимое элемента div. В данном случае в качестве содержимого выступает простой текст “Текст элемента div”.

Элементы также могут состоять из одного тега, например, элемент
, функция которого - перенос строки.

**<div>Текст
 элемента div</div>**

Такие элементы еще называют пустыми элементами (void elements) [1]. Помимо этого, в некоторых тегах наличие закрывающих слешей необязательно. К таким тегам относится тег перехода на новую строку:
.

Каждый элемент внутри открывающего тега может иметь атрибуты. Атрибуты позволяют изменить свойства тега, заданные по умолчанию.

Например:

<div style="color: blue;">Кнопка</div>

<input type="button" value="Нажать">

В приведенном примере определено два элемента: div и input. Элемент div имеет атрибут style. После знака равно в кавычках пишется значение атрибута: style="color: red;";, которое указывает, что цвет текста будет красным.

Второй элемент - элемент input, состоящий из одного тега, имеет два атрибута: type (указывает на тип элемента - кнопка) и value (определяет текст на кнопке).

Существуют глобальные или общие для всех элементов атрибуты, как например, style, а есть специфические, применяемые к определенным элементам, как например, type.

Кроме обычных атрибутов существуют еще булевы или логические атрибуты (boolean attributes). Подобные атрибуты могут не иметь значения. Например, у кнопки можно задать атрибут disabled:

<input type="button" value="Нажать" disabled>

Атрибут disabled указывает, что данный элемент отключен [3].

1.2.1 Глобальные атрибуты

В HTML5 есть набор глобальных атрибутов, которые применимы к любому элементу HTML5:

- accesskey: определяет клавишу для быстрого доступа к элементу;
- class: задает класс CSS, который будет применяться к элементу;
- contenteditable: определяет, можно ли редактировать содержимое элемента;
- contextmenu: определяет контекстное меню для элемента, которое отображается при нажатии на элемент правой кнопкой мыши;
- dir: устанавливает направление текста в элементе;
- draggable: определяет, можно ли перетаскивать элемент;
- dropzone: определяет, можно ли копировать переносимые данные при переносе на элемент;
- hidden: скрывает элемент;

- id: уникальный идентификатор элемента. На веб-странице элементы не должны иметь повторяющихся идентификаторов;
- lang: определяет язык элемента;
- spellcheck: указывает, будет ли для данного элемента использоваться проверка правописания;
- style: задает стиль элемента;
- title: устанавливает дополнительное описание для элемента;
- translate: определяет, должно ли переводиться содержимое элемента.

Но, как правило, из всего этого списка наиболее часто используются три: class, id и style [4].

1.2.2 Пользовательские атрибуты

В отличие от предыдущей версии языка разметки, в HTML5 были добавлены пользовательские атрибуты (custom attributes). Теперь разработчик веб-страницы сам может определить любой атрибут, предваряя его префиксом data-. Например:

```
<input type="button" value="Нажать" data-color="blue" >
```

Здесь определен атрибут data-color, который имеет значение “blue”. Хотя для этого элемента, в html не существует подобного атрибута. Он определяется разработчиком и устанавливается у него любое значение.

1.2.3 Одинарные и двойные кавычки

Нередко можно встретить случаи, когда в HTML при определении значений атрибутов применяются как одинарные, так и двойные кавычки. Например:

```
<input type='button' value='Нажать'>
```

И одинарные, и двойные кавычки в данном случае допустимы, хотя чаще применяются двойные кавычки. Однако иногда само значение атрибута может содержать двойные кавычки, и в этом случае все значение лучше поместить в одинарные:

`<input type="button" value='Кнопка "Привет мир"'>`

1.3 Создание документа HTML5

Элементы являются кирпичиками, из которых складывается документ html5. Для создания документа необходимо создать простой текстовый файл, назвать index, а в качестве расширения файла указать *.html (рисунок 1.1).

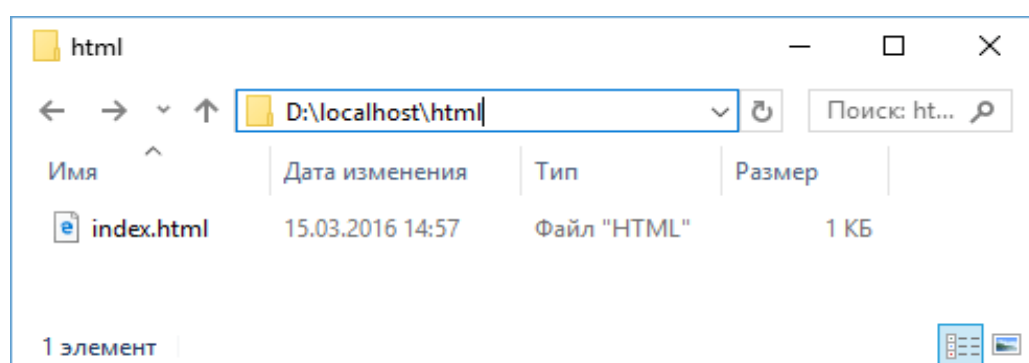


Рисунок 1.1 - Структура документа HTML5

Затем можно открыть этот файл в любом текстовом редакторе, например, в Notepad++. Добавить в файл следующий текст:

`<!DOCTYPE html>`

`<html>`

`...`

`</html>`

Для создания документа HTML5 необходимо в первую очередь два элемента: DOCTYPE и html. Элемент doctype или Document Type Declaration сообщает веб-браузеру тип документа. `<!DOCTYPE html>` указывает, что данный документ является документом html и что используется html5, а не html4 или какая-то другая версия языка разметки.

Элемент HTML между своим открывающим и закрывающим тегами содержит все содержимое документа. Внутри элемента HTML можно разместить два других

элемента: HEAD и BODY. Элемент HEAD содержит метаданные веб-страницы - заголовок веб-страницы, тип кодировки и т.д., а также ссылки на внешние ресурсы - стили, скрипты, если они используются. Элемент BODY, собственно, определяет содержимое HTML-страницы.

Изменим содержимое файла index.html следующим образом:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Страница HTML5</title>
  </head>
  <body>
    <div>Добро пожаловать! (Контент страницы HTML5)</div>
  </body>
</html>
```

В элементе head определено два элемента:

- элемент title, который представляет заголовок страницы;
- элемент meta, который определяет метаинформацию страницы.

Для корректного отображения символов предпочтительно указывать кодировку. В данном случае с помощью атрибута charset="utf-8" указана кодировка utf-8. В пределах элемента body используется только один элемент - div, который оформляет блок. Содержимым этого блока является простая строка. Поскольку выбрана в качестве кодировки utf-8, то браузер будет отображать веб-страницу именно в этой кодировке (рисунок 1.2).

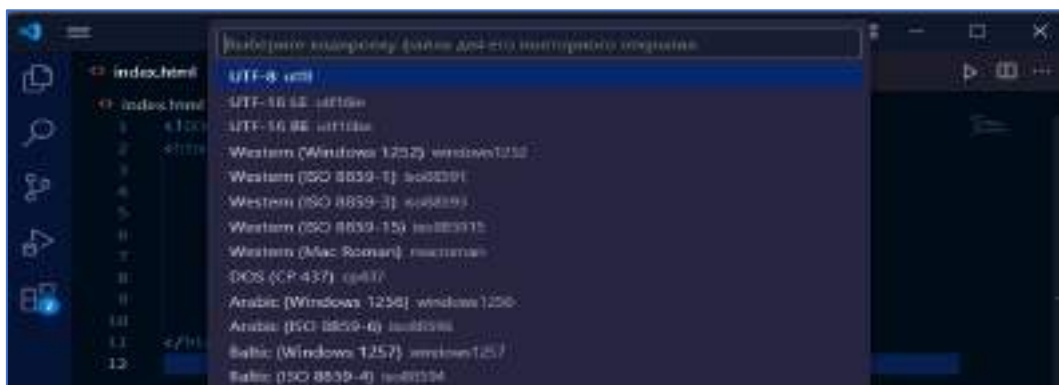


Рисунок 1.2 - Установка кодировки html-файла в VS Code

После этого в статусной строке будет можно будет увидеть UTF-8 w/o BOM, что будет указывать, что нужная кодировка установлена.

Далее следует сохранить и открыть файл index.html в браузере (рисунок 1.3).

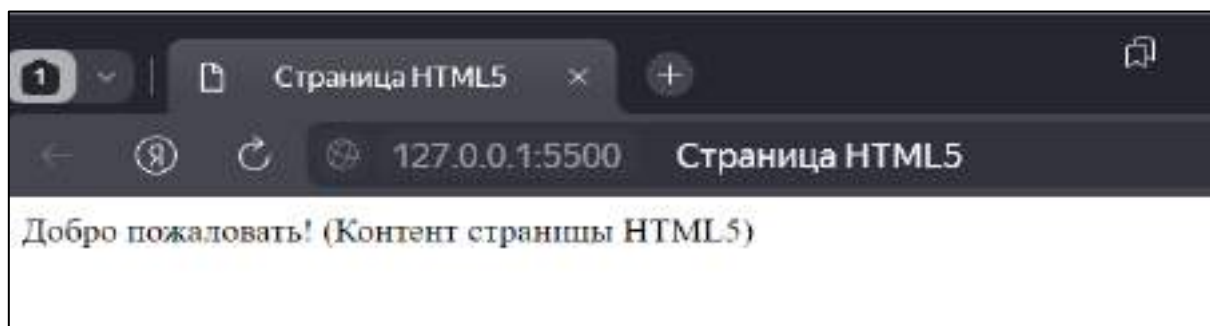


Рисунок 1.3 - Документ HTML5

Таким образом, создан первый документ HTML5. Так как указан в элементе title заголовок “Документ HTML5”, то именно такое название будет иметь вкладка браузера [1].

Так как указана кодировка utf-8, то веб-браузер будет корректно отображать кириллические символы. А весь текст, определенный внутри элемента body, можно увидеть в основном поле браузера.

Все, что видно в своем браузере при загрузке веб-страницы, основная часть документа html располагается между тегами <body> и </body>. Здесь размещаются большинство элементов html.

1.4 Элемент head и метаданные веб-страницы

Как правило, одним из первых элементов html-документа является элемент head, задача которого состоит в установке метаданных страницы и ряда сопроводительной информации. Метаданные содержат информацию о html-документе. Помимо метаданных заголовков документа может содержать информацию об стилях страницы, заданных с помощью различных селекторов.

1.4.1 Заголовок

Для установки заголовка документа, который отображается на вкладке браузера, используется элемент title.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Заголовок - title</title>
  </head>
  <body> <p>Контент страницы HTML5</p></body>
</html>
```

Реализация данного кода представлена на рисунке 1.4.

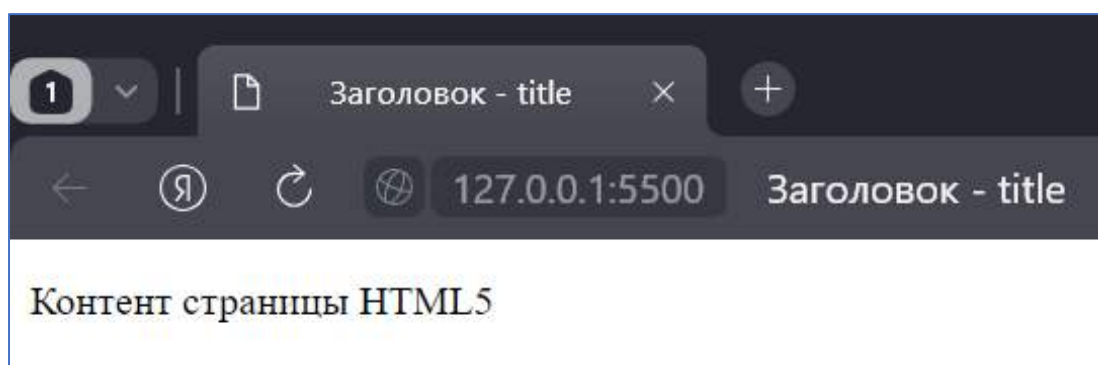


Рисунок 1.4 – Использование заголовков в html

1.4.2 Элемент base

Элемент base позволяет указать базовый адрес, относительно которого устанавливаются другие адреса, используемые в документе:

```
<!DOCTYPE html>
<html> <head>
  <base href="content/">
  <meta charset="utf-8">
  <title>Элемент base</title> </head>
  <body> <a href="newpage.html">Перейти</a> </body>
</html>
```

Хотя для ссылки в качестве адреса указана страница newpage.html, но фактически ее адресом будет content/newpage.html. То есть в одной папке с текущей страницей должна быть подпапка content, в которой должен находиться файл newpage.html

Можно также указывать полный адрес: **<base href="http://www.microsoft.com/">**. В этом случае ссылка будет вести на сайт <http://www.microsoft.com/newpage.html>.

1.4.3 Элемент meta

Элемент meta определяет метаданные документа. Чтобы документ корректно отображал текст, необходимо задать кодировку с помощью атрибута charset. Рекомендуемой кодировкой является utf-8:

<meta charset="utf-8">

При этом надо помнить, что указанная элементе meta кодировка должна совпадать с кодировкой самого документа. Как правило, текстовый редактор позволяет указать кодировку документа. Если ориентироваться на utf-8, то в настройках текстового редактора надо выбирать UTF-8.

Элемент meta также имеет два атрибута: name и content. Атрибут name содержит имя метаданных, а content - их значение.

По умолчанию в HTML определены пять типов метаданных:

- application name: название веб-приложения, частью которого является данный документ;
- author: автор документа;
- description: краткое описание документа;
- generator: название программы, которая сгенерировала данный документ;
- keywords: ключевые слова документа.

Надо отметить, что наиболее актуальным является тип description. Его значение поисковики часто используют в качестве аннотации к документу в поисковой выдаче. Далее представлен документ с использованием элементов meta.


```
<!DOCTYPE html>
```

```
<html> <head>
```

```
    <meta charset="utf-8">
```

```
    <base href="content/">
```

```
    <title>Заголовок - title</title>
```

```
    <meta name="contents" content="Первая страница HTML5">
```

```
    <meta name="author" content="Tagirova L.F. ">
```

```
</head>
```

```
<body> <a href="pagecontent.html">Контент страницы HTML5</a>
```

```
</body> </html>
```

Далее в документ добавлен элемент meta descripton.

```
<!DOCTYPE html>
```

```
<html> <head>
```

```
    <meta charset="utf-8">
```

```
    <base href="content/">
```

```
    <title>Заголовок - title</title>
```

```
    <meta name="contents" description="Контент страницы HTML">
```

```
</head> <body> <a href="pagecontent.html">Контент страницы HTML5</a>
```

```
</body> </html>
```

1.5 Элементы группировки в HTML5

1.5.1 Элемент div

Элемент div служит для структуризации контента на веб-странице, для заключения содержимого в отдельные блоки [6]. Div создает блок, который по умолчанию растягивается по всей ширине браузера, а следующий после div элемент переносится на новую строку. Например:

```
<body>
```

```
    <div>Заголовок контента HTML5</div>
```

```
<div>Контент страницы HTML5</div>
</body>
```

Реализация данного кода представлена на рисунке 1.5.

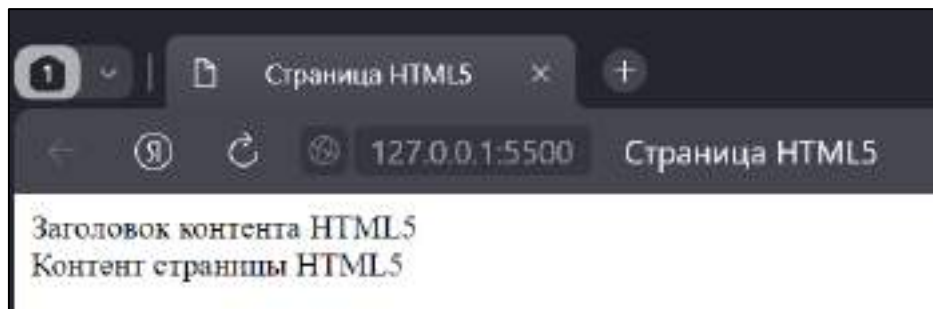


Рисунок 1.5 - Блоки DIV в HTML5

1.5.2 Параграфы

Параграфы создаются с помощью тегов `<p>` и `</p>`, в которые заключают некоторое содержимое. Каждый новый параграф располагается на новой строке. Далее представлен пример применением параграфов [7]:

```
<body>
  <div>Заголовок контента HTML5</div>
  <div> <p>Параграф 1</p>
    <p>Параграф 2</p> </div> </body>
```

Реализация данного кода представлена на рисунке 1.6.

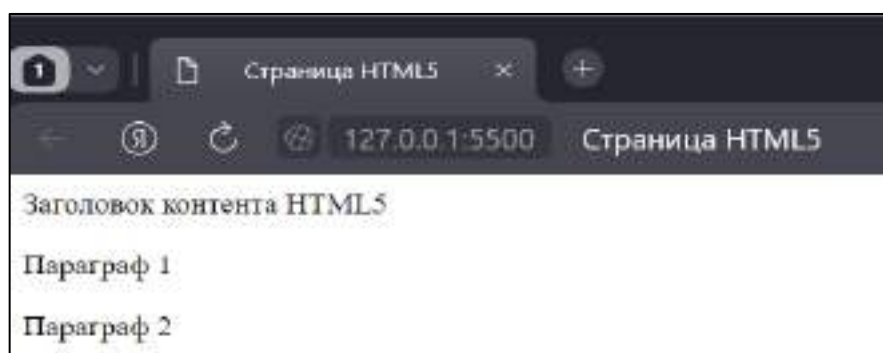


Рисунок 1.6 - Параграфы в html5

Если в рамках одного параграфа необходимо перенести текст на другую строку, то можно воспользоваться элементом `
`:

**<p>Первая строка.
 Вторая строка. </p>**

1.5.3 Элемент pre

Элемент pre выводит предварительно отформатированный текст так, как он определен.

```
<body> <pre> Михаил Лермонтов  
    Один из самых известных русских поэтов  
        признание к нему пришло еще при жизни. </pre> </body>
```

Реализация данного кода представлена на рисунке 1.7.

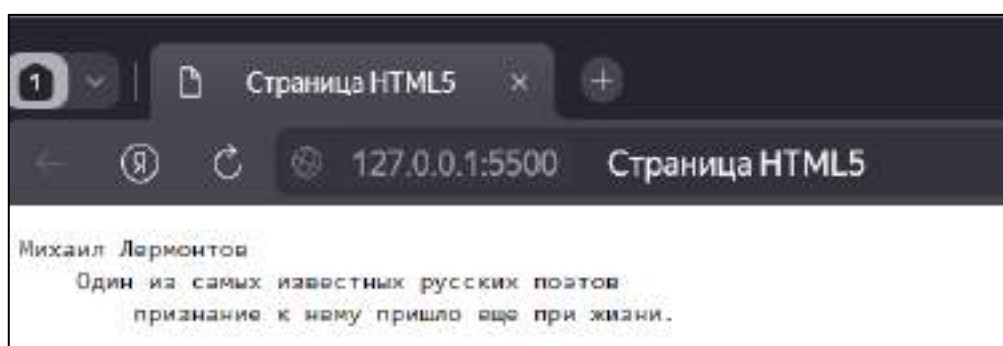


Рисунок 1.7 - Элемент pre в HTML5

1.5.4 Элемент span

Элемент span обтекает некоторый текст по всей его длине и служит преимущественно для стилизации заключенного в него текстового содержимого. В отличие от блоков div или параграфов span не переносит содержимое на следующую строку.

```
<body>  
    <div>Заголовок контента HTML5</div>  
    <div>  
        <p><span style="color: blue;">Параграф</span> 1</p>  
        <p> <span>Параграф</span> 2</p>  
    </div> </body>
```

Реализация данного кода представлена на рисунке 1.8.

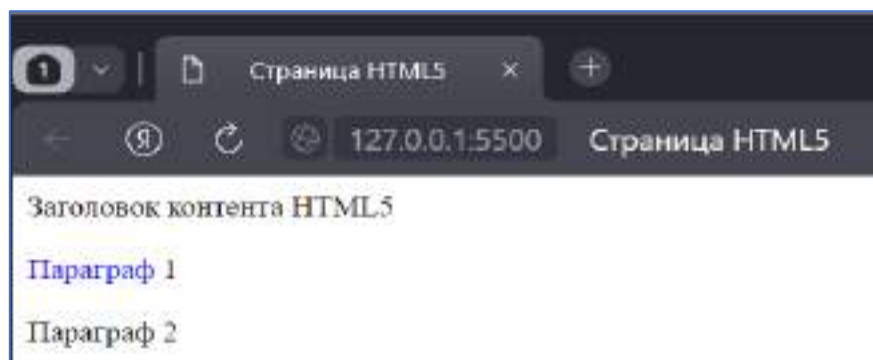


Рисунок 1.8 - Элемент span в html5

При этом, сам по себе span ничего не делает. Так, во втором параграфе span не повлиял на внутренне текстовое содержимое. А в первом параграфе элемент span содержит атрибут стиля: `style="color: red: blue;"`, который устанавливает для текста синий цвет. При этом элементы `div` и `p` являются блочными, элемент `div` может содержать любые другие элементы, а элемент `p` - только строчные элементы. В отличие от них элемент `span` является строчным, встраивает свое содержимое во внешний контейнер - тот же `div` или параграф. Не стоит помещать блочные элементы в строчный элемент `span`.

1.6 Заголовки в HTML5

Элементы `<h1>`, `<h2>`, `<h3>`, `<h4>`, `<h5>` и `<h6>` используются для создания заголовков различных уровней (рисунок 1.10):

```
<body>
  <h1>1-ый уровень</h1>
  <h2>2-ой уровень </h2>
  <h3>3-ий уровень </h3>
  <h4>4-ый уровень </h4>
  <h5>5-ый уровень </h5>
  <h6>6-ой уровень </h6>
</body>
```

Реализация данного кода представлена на рисунке 1.9.

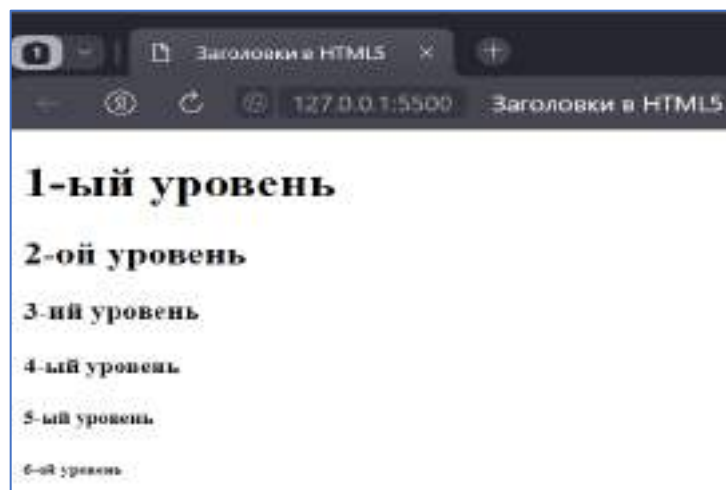


Рисунок 1.9 - Заголовки в HTML5

Заголовки выделяются жирным шрифтом и имеют определенный размер. При определении заголовков следует учитывать, что на странице должен быть только один заголовок верхнего уровня. Он функционирует как основной заголовок веб-страницы.

1.7 Форматирование текста

Ряд элементов HTML используется для форматирования текстового содержимого, например, для выделения жирным или курсивом и т.д. Далее описаны эти элементы:

- ``: выделяет текст жирным;
- ``: зачеркивает текст;
- `<i>`: выделение текста курсивом;
- ``: выделяет текст курсивом, в отличие от тега `<i>` носит логическое значение, придает выделяемому тексту оттенок важности;
- `<s>`: зачеркивает текст;
- `<small>`: делает текст чуть меньше размером, чем окружающий;

-: выделяет текст жирным. В отличие от тега предназначен для логического выделения, чтобы показать важность текста. А не носит характера логического выделения, выполняет функции только форматирования;

-<sub>: помещает текст под строкой;

-<sup>: помещает текст над строкой;

-<u>: подчеркивает текст;

-<mark>: выделяет текст цветом, придавая ему оттенок важности.

Далее представлен пример применения всех этих элементов.

```
<body>  <p>Форматирование в <mark>HTML5</mark></p>
<p> <b>Выделенный</b> контент</p>
<p><strong>Важный</strong> контент</p>
<p><del>Зачеркнутый</del> контент</p>
<p> <s>Недействительный</s> контент</p>
<p> <em>Важный</em> контент</p>
<p>Контент выделенный <i>курсивом</i> </p>
<p> <ins>Добавленный</ins> контент</p>
<p> <u>Подчеркнутый</u> контент</p>
<p>z<sub>i</sub> = y<sup><small>3</small></sup> + x<sup><small>3</small></sup> </p>
</body>
```

Реализация данного кода представлена на рисунке 1.10.

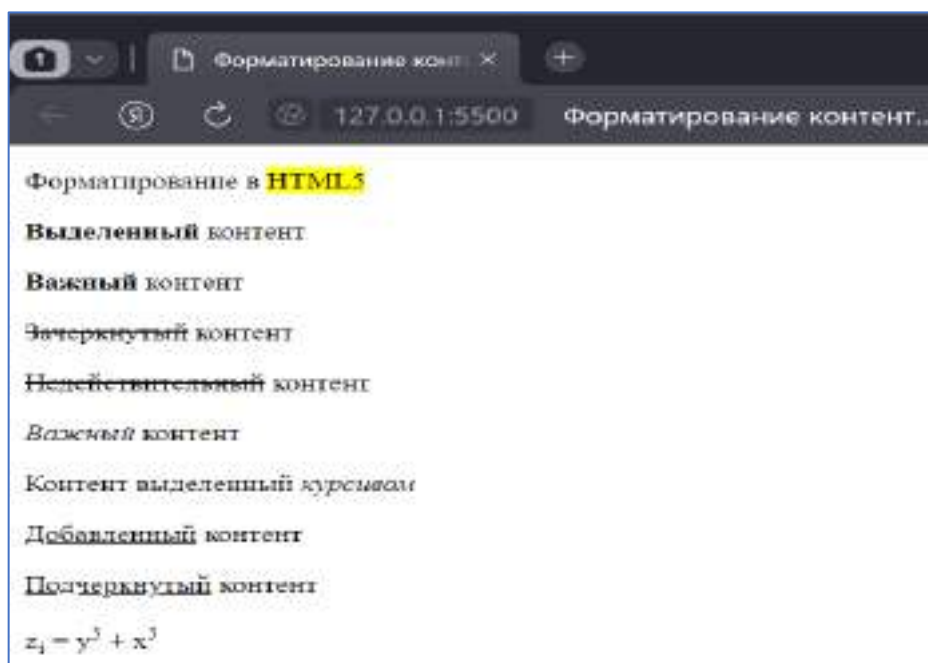


Рисунок 1.10 - Использование тегов форматирования текста

1.8 Escape-последовательности в HTML

В HTML можно использовать экранирующий символ (escape-последовательность) для представления любого символа Unicode, используя только буквы ASCII [8].

Экранирующие символы очень полезны для представления символов, которые не являются очевидными или неоднозначны, в частности, это относится к символам «<», «>», «&», «_», «©». В таблице 1.1 приведен список escape-последовательностей в HTML.

Таблица 1.1 - Некоторые популярные экранирующие символы

Вид	Обозначение	Десятичный код	Название символа
	 	 	неразрывный пробел
<	<	<	левая угловая скобка
>	>	>	правая угловая скобка
¢	¢	¢	цент
£	£	£	фунт стерлингов
¥	¥	¥	йена
§	§	§	параграф
©	©	©	знак авторского права
®	®	®	охраняемый знак
«	«	«	открывающая двойная угловая кавычка
»	»	»	закрывающая двойная угловая кавычка
°	°	°	градус
¿	¿	¿	перевернутый вопросительный знак
μ	µ	µ	знак микро
¼	¼	¼	одна четвертая
½	½	½	одна вторая

Продолжение таблицы 1.1

¾	¾	¾	три четверти
&	&	&	амперсант
€	€	€	евро
↔	↔	↔	стрелка влево-вправо
⇔	⇔	⇔	двойная стрелка влево-вправо
∞	∞	∞	бесконечность
"	″	″	знак двойной прим
¶	¶	¶	конец абзаца

В приведенном ниже примере представлена HTML-страница, на которой использованы различные экранирующие символы.

```
<body>
```

```
<h3> Использование Escape-последовательностей в HTML </h3>
```

```
<div> &copy; &laquo;Теги HTML - это команды, которые говорят браузеру, что и в  
каком порядке показывать на экране.
```

```
</div>
```

```
<div> У каждого тега есть имя, которое расположено в угловых скобках.
```

```
</div>
```

```
<div> Самая простая HTML-страница состоит из трех тегов: &lt;html&gt;, &lt;head&gt;  
и &lt;body&gt;&raquo; </div><div>
```

```
</div>
```

```
</body>
```

Реализация данного кода представлена на рисунке 1.11.

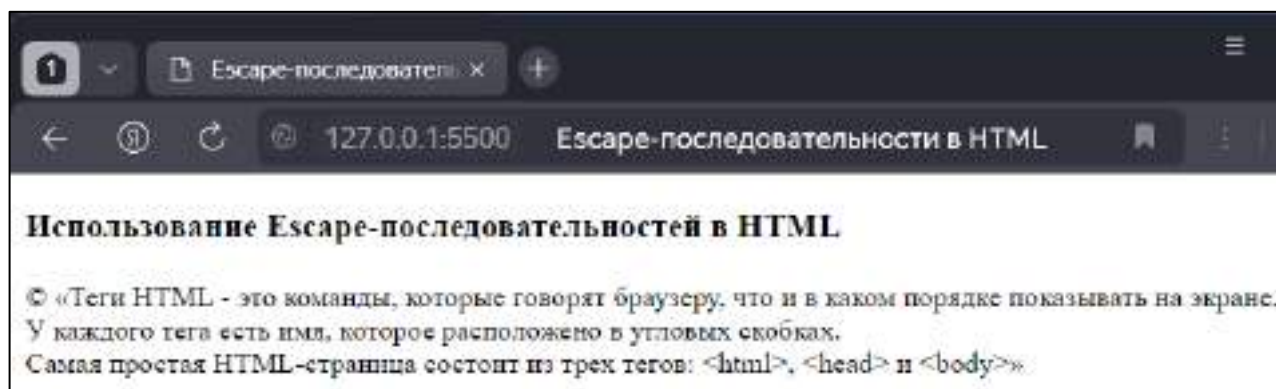


Рисунок 1.11 – Использование Escape-последовательностей в HTML

1.9 Работа с изображениями

Для вывода изображений в HTML используется элемент `img`. Этот элемент представляет два важных атрибута:

- `src`: путь к изображению. Это может быть относительный или абсолютный путь в файловой системе или адрес в интернете;

- `alt`: текстовое описание изображения. Если браузер не может отобразить изображение, то он показывает вместо самой картинки данное текстовое описание.

Пример работы с рисунками представлен в нижеследующем примере.

```
<body>
```

```

```

```
</body>
```

Реализация данного кода представлена на рисунке 1.12.

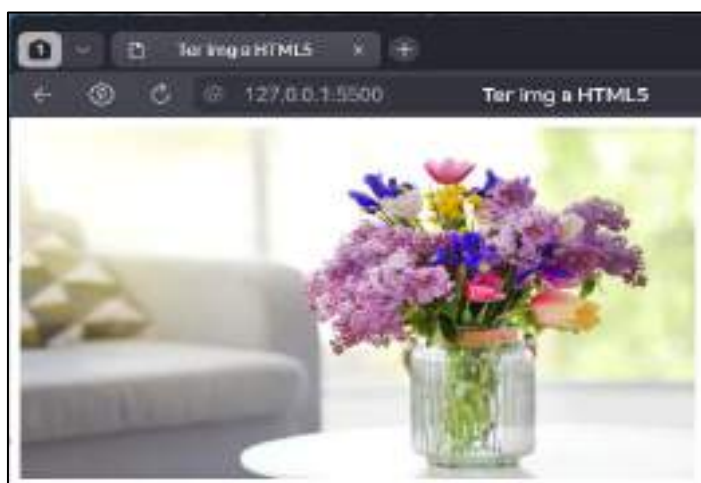


Рисунок 1.12 - Тег `img` в HTML5

В данном примере файл изображения `flowerss.jpg` находится в одной папке с веб-страницей `index.html`. При этом рисунок задан в исходном размере, расстояние до окружающего текста и обтекание по умолчанию.

Для того, чтобы изменить значения рисунка по умолчанию у тега `img` можно применять дополнительные классические атрибуты:

- `height` – изменение высоты рисунка (задается по умолчанию в пикселях),

- width – изменение ширины рисунка (задается по умолчанию в пикселях),
- usemap – ссылка на элемент <map>, содержащий координаты для клиентской карты-изображения,
- hspace – расстояние от рисунка до окружающего текста по горизонтали,
- vspace – расстояние от рисунка до окружающего текста по вертикали,
- border – рамка.

HTML5 предлагает современные настройки для работы с рисунками.

1) Атрибут srcset. Он представляет собой способ предложить браузеру версию картинки с повышенным разрешением:

```

```

Также атрибут srcset можно использовать в сочетании с атрибутом sizes, чтобы подсказать браузеру, какие варианты картинок есть, и помочь ему выбрать подходящие ситуации.

2) Атрибут sizes. HTML даёт возможность загружать разные изображения в зависимости от разных условий. Частая ситуация: разные картинки под разные ширины экранов. Можно делать это при помощи тега <picture>, а можно задать атрибуты srcset и sizes прямо в теге .

- srcset - атрибут, принимающий несколько строк, разделённых запятой. Каждая строка должна содержать ссылку на картинку и указание фактической ширины картинки, значения разделяются пробелом. При этом после ширины ставится единица измерения w, а не px.

- sizes - атрибут, в котором указывается одно или несколько условий через запятую. Каждая строка состоит из медиавыражения и ширины блока для картинки, разделённых пробелом. Ширину блока для картинки можно указывать в любых единицах измерения, кроме процентов.

```

```

Далее приведен пример, в котором использовался тот же рисунок, однако были изменены его размер, добавлено обтекание и рамка.

```
<body> <img src=flowerss.jpg alt="Цветы в вазе на подоконнике" style="float: left; margin-right:10px; width:250px; height:150px; border=2px;" />
```

```
<h2>Флористка</h2> <p>Флористика - это искусство использовать растительные  
элементы для создания композиций или украшения интерьера/экстерьера. Профессию  
флориста нередко сравнивают с художником, только вместо красок мастер использует  
цветочные бутоны, зеленые растения, природные материалы.</p> <p>Создание букета  
требует определённых знаний. В букете нужно соединить цветы и зелень, которые будут  
дополнять друг друга, не конфликтуя между собой. Только так можно добиться единого  
шедевра. Специалисты флористики создали настоящую науку по комбинированию бутонов в  
букете. Важно учитывать не только цвет и другие параметры, но и природные данные цветов.  
</p> </body>
```

Реализация данного кода представлена на рисунке 1.13.

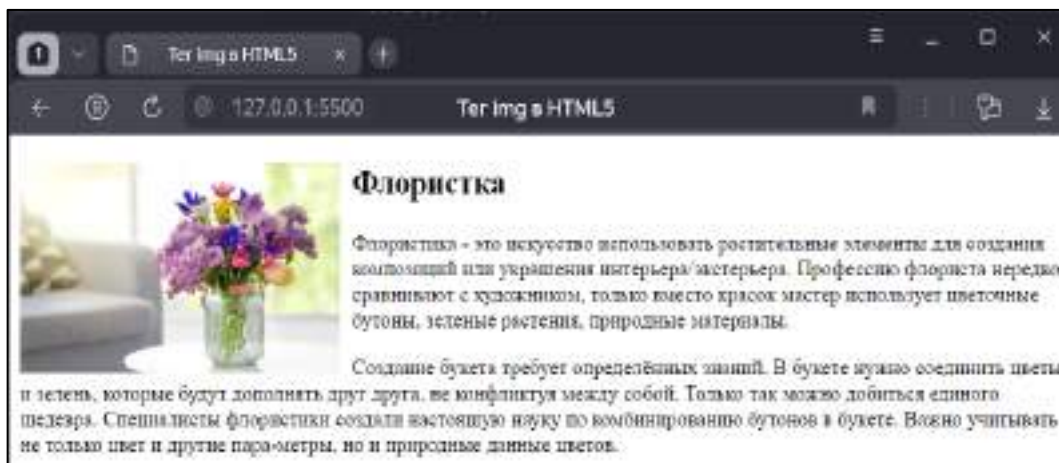


Рисунок 1.13 – Использование дополнительных атрибутов тега `img`

1.10 Мультимедиа

1.10.1 Видео

Для воспроизведения видео в HTML5 используется элемент `video`. Чтобы настроить данный элемент, можно использовать следующие его атрибуты:

- src: источник видео, это может быть какой-либо видеофайл;
- width: ширина элемента;
- height: высота элемента;
- controls: добавляет элементы управления воспроизведением;
- autoplay: устанавливает автовоспроизведение;
- loop: задает повторение видео;
- muted: отключает звук по умолчанию.

Хотя можно установить ширину и высоту, но они не окажут никакого влияния на отношения ширины и высоты самого видео. Например, если видео имеет формат 375×240, то, к примеру, при настройках width="375" height="280" видео будет центрироваться на 280-пиксельном пространстве в HTML.

```
<body> <video src="flower.mp4" width="400" height="300" controls ></video></body>
</html>
```

Реализация данного кода представлена на рисунке 1.14.



Рисунок 1.14 - Видео в HTML5

Далее были применены атрибуты autoplay и loop:

```
<video src="flower.mp4" width="400" height="300" controls autoplay loop>
</video>
```

Теперь видео будет автоматически проигрываться бесконечное число раз.

Если при воспроизведении необходимо отключить звук, то можно воспользоваться атрибутом muted:

```
<video src="flower.mp4" width="400" height="300" controls muted></video>
```

1.10.1.1 Атрибут preload

Еще один атрибут - preload призван управлять загрузкой видео. Он принимает следующие значения:

- auto: видео и связанные с ним метаданные будут загружаться до того, как видео начнет воспроизводиться;
- none: видео не будет загружаться в фоне, пока пользователь не нажмет на кнопку начала проигрывания;
- metadata: в фоне до воспроизведения будут загружаться только метаданные (данные о формате, длительности и т.д), само видео не загружается.

```
<video      src="flower.mp4"      width="400"      height="300"      controls  
preload="auto"></video>
```

1.10.1.2 Атрибут poster

Атрибут poster позволяет установить изображение, которое будет отображаться до запуска видео. Этому атрибуту в качестве значения передается путь к изображению:

```
<video      src="flower.mp4"      width="400"      height="300"      controls  
poster="mycat.jpg">  
</video>
```

1.10.1.3 Поддержка форматов видео

Главной проблемой при использовании элемента video является поддержка различными веб-браузерами определенных форматов. С помощью вложенных

элементов source можно задать несколько источников видео, один из которых будет использоваться:

```
<video width="400" height="300" controls>  
  <source src="flower.mp4" type="video/mp4">  
  <source src="flower.webm" type="video/webm">  
  <source src="flower.ogv" type="video/ogg">  
</video>
```

Элемент source использует два атрибута для установки источника видео:

- src: путь к видеофайлу;
- type: тип видео (MIME-тип).

Если браузер не поддерживает первый тип видео, то он пытается загрузить второй видеофайл. Если же и тип второго видеофайла не поддерживается, то браузер обращается к третьему видеофайлу.

1.10.2 Аудио

Для воспроизведения звука без видео в HTML5 применяется элемент audio. Он во многом похож на элемент video. Для настройки элемента audio можно использовать следующие его атрибуты:

- src: путь к аудиофайлу;
- controls: добавляет элементы управления воспроизведением;
- autoplay: устанавливает авто воспроизведение;
- loop: задает повторение аудиофайла;
- muted: отключает звук по умолчанию;
- preload: устанавливает режим загрузки файла.

Действие всех этих атрибутов будет аналогично их действию в элементе video.

```
<body>  
  <audio src="sound.mp3" controls>  
</audio>  
</body>
```

Реализация данного кода представлена на рисунке 1.15.

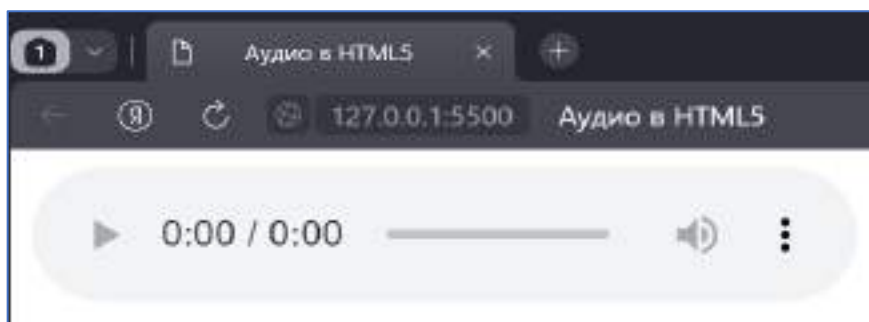


Рисунок 1.15 - Аудио в HTML5

Ключевым моментом для работы с аудио является поддержка браузером тех или иных форматов. На данный момент подавляющее большинство браузеров поддерживают mp3. Однако если есть неуверенность, что аудио в определенном формате будет поддерживаться браузером пользователя, то можно использовать вложенный элемент source и указать аудио в иных форматах:

```
<audio width="400" height="300" controls>  
  <source src="sound.mp3" type="audio/mpeg">  
  <source src="sound.m4a" type="audio/aac">  
  <source src="sound.ogg" type="audio/ogg">  
</audio>
```

Как и в случае с элементом video, здесь у элемента source устанавливается атрибут src с ссылкой на файл и атрибут type - тип файла.

1.11 Списки в HTML

Для создания списков в HTML5 применяются элементы `` (нумерованный список) и `` (нечисловый список). В нумерованном списке для нумерации элементов по умолчанию используется стандартные цифры от 1. В нечисловом списке каждый элемент обозначается закрашенным диском.

Далее представлен пример использования этих тегов.

```
<body> <h2> Нумерованный список</h2>
```

```

<ol> <li>Розы</li> <li>Хризантемы</li> <li>Тюльпаны</li> <li>Гладиолусы</li> </ol>
<h2> Ненумерованный список</h2>
<ul><li>Пионы</li> <li>Лилии</li> <li>Герберы</li> <li>Ромашки</li>
</ul>
</body>

```

Реализация данного кода представлена на рисунке 1.16.

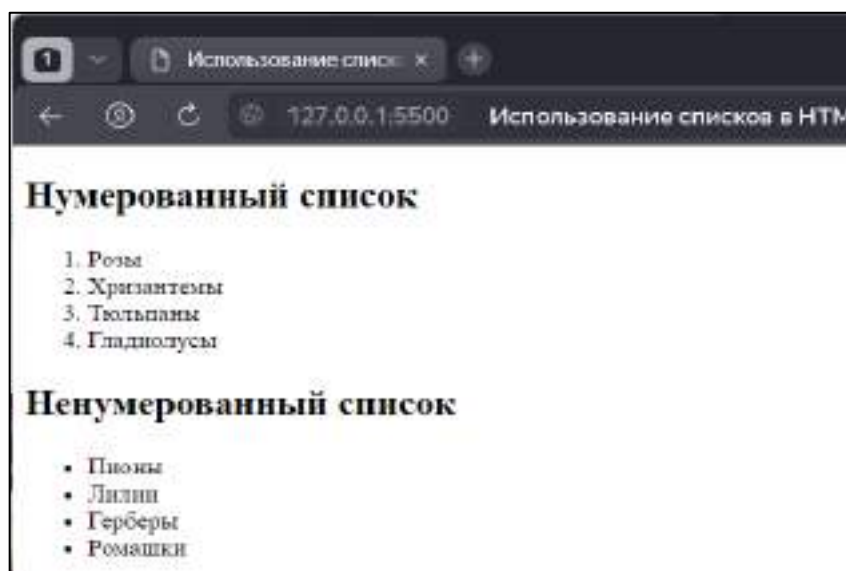


Рисунок 1.16 – Использование списков в HTML5

При необходимости можно настроить нумерацию или отражаемый рядом с элементом символ с помощью стиля `list-style-type`. Данный стиль может принимать множество различных значений. Для нумерованных списков стиль `list-style-type` может принимать следующие значения:

- `decimal`: десятичные числа, отсчет идет от 1;
- `decimal-leading-zero`: десятичные числа, которые предваряются нулем, например, 01, 02, 03, ... 98, 99;
- `lower-roman`: строчные римские цифры, например, i, ii, iii, iv, v,...;
- `upper-roman`: заглавные римские цифры, например, I, II, III, IV, V,...;
- `lower-alpha`: строчные римские буквы, например, a, b, c,... z;
- `upper-alpha`: заглавные римские буквы, например, A, B, C, ... Z.

Для нумерованных список с помощью атрибута `start` можно дополнительно задать символ, с которого будет начинаться нумерация.

Далее представлен пример работы с атрибутами нумерованных списков для вывода наименований пород кошек.

```
<h2>list-style-type = decimal</h2><ol style="list-style-type: decimal;" start="3">  
  <li>Сиамская</li> <li>Британская </li> <li>Мейн-кун</li></ol>  
<h2>list-style-type = upper-roman</h2><ol style="list-style-type: upper-roman;">  
  <li>Персидская </li><li>Русская голубая </li> <li> Сфинкс </li></ol>  
<h2>list-style-type = lower-alpha</h2><ol style="list-style-type: lower-alpha;">  
  <li> Тонкинская</li><li>Шиншилла</li> <li>Сибирская </li></ol>
```

Реализация данного кода представлена на рисунке 1.17.

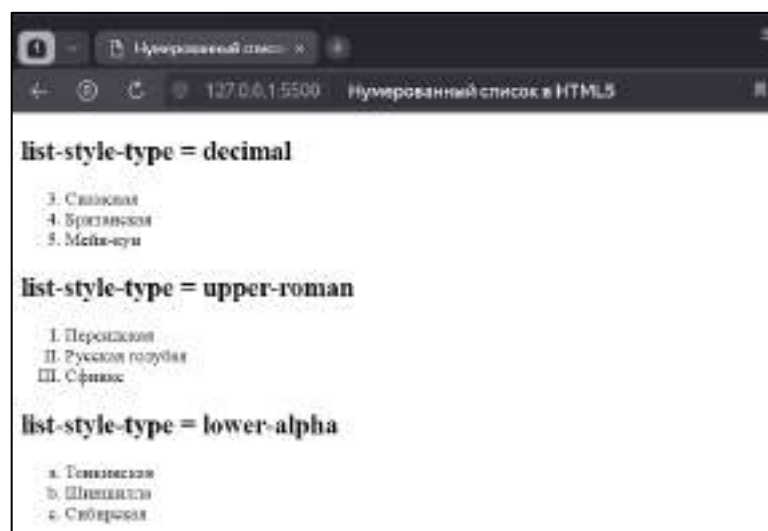


Рисунок 1.17 - Нумерованный список в HTML5

Для ненумерованного списка атрибут list-style-type может принимать следующие значения:

- disc: черный диск;
- circle: пустой кружочек;
- square: черный квадратик.

Далее представлен пример работы с нумерованными списками для вывода наименований пород собак.

```
<h2>list-style-type = disc</h2><ul style="list-style-type: disc;">  
<li>Австралийская овчарка</li><li>Алабай</li><li>Спаниель</li></ul>  
<h2>list-style-type = circle</h2><ul style="list-style-type: circle;">  
<li>Бигль</li> <li>Боксер</li> <li>Болонка </li></ul>
```

```
<h2>list-style-type = square</h2><ul style="list-style-type: square;">  
<li> Корги</li><li>Чихуахуа</li> <li>Доберман </li></ul>
```

Реализация данного кода представлена на рисунке 1.18.

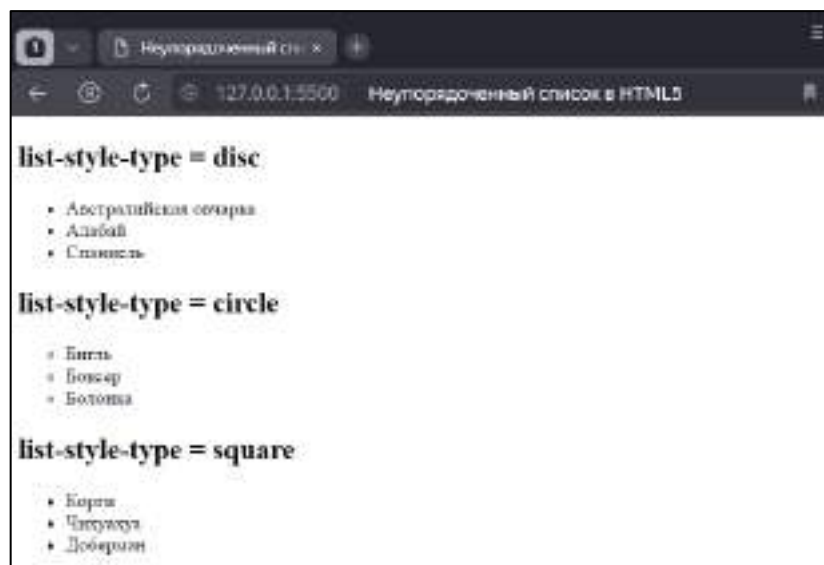


Рисунок 1.18 - Неупорядоченный список в HTML5

Еще одну интересную возможность по настройке списков предоставляет стиль list-style-image. Он задает изображение, которое будет отображаться рядом с элементом списка.

```
<ul style="list-style-image: url(/birds.png)">  
<li>Лебедь</li> <li>Орел</li>  
<li>Соловей</li> <li>Павлин</li></ul>
```

Реализация данного кода представлена на рисунке 1.19.

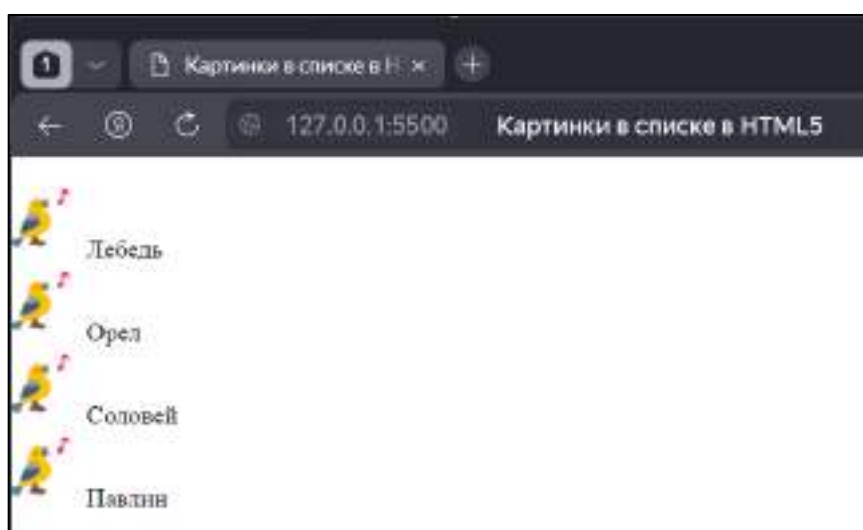


Рисунок 1.19 - Картинки в списке в HTML5

Стиль `list-style-image` в качестве значения принимает `url(/birds.png)`, где “birds” - это название файла изображения, который находится в одной папке с веб-страницей `index.html`.

Одним из распространенных способов стилизации списков представляет создание горизонтального списка. Для этого для всех элементов списка надо установить стиль `display:inline`.

```
<head> <style>  ul#menu li {display: inline;} </style> </head>
<body> <ul id="menu">
    <li>Пушкин</li> <li>Лермонтов</li><li>Пришвин</li><li>Маяковский</li> </ul>
</body> </html>
```

Реализация данного кода представлена на рисунке 1.20.

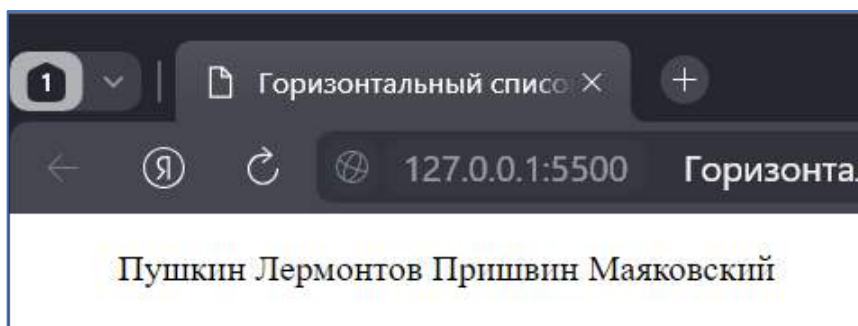


Рисунок 1.20 - Горизонтальный список

1.12 Элемент details

Элемент `details` позволяет создавать раскрываемый блок, который по умолчанию скрыт. Иногда такой элемент еще называют `accordion`. Данный элемент содержит элемент `summary`, который представляет заголовок для блока, и этот заголовок отображается в скрытом режиме. Далее представлен пример использования аккордиона.

```
<body> <details> <summary>Поэты</summary>
<ul> <li>Пушкин</li>
    <li>Лермонтов</li>
```

```
<li>Тютчев</li>
<li>Блок</li>
</ul>
</details>
</body>
```

По умолчанию видно только заголовок summary (рисунок 1.21).

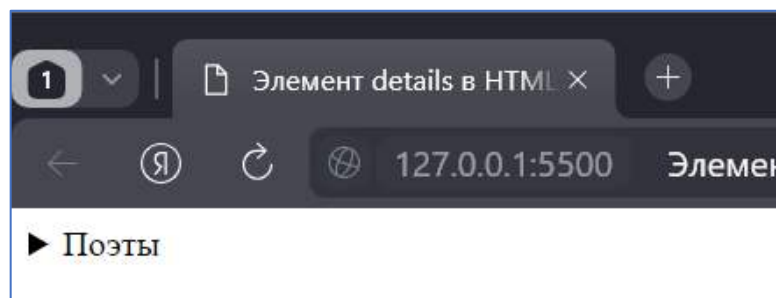


Рисунок 1.21 - Элемент summary в HTML5

Нажав на стрелку или заголовок, можно раскрыть блок (рисунок 1.22).

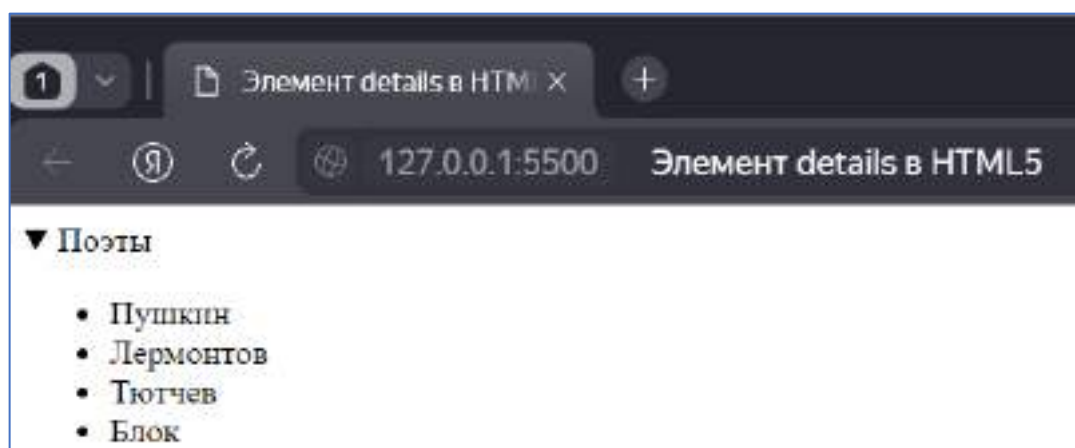


Рисунок 1.22 - Элемент details в HTML5

1.13 Обозначение цветов в HTML

В HTML используется цветовая модель RGB (Red, Green, Blue) цвета представляют тремя значениями, описывающими цветовые каналы - соответственно - красный, зелёный и синий канал (рисунок 1.23).

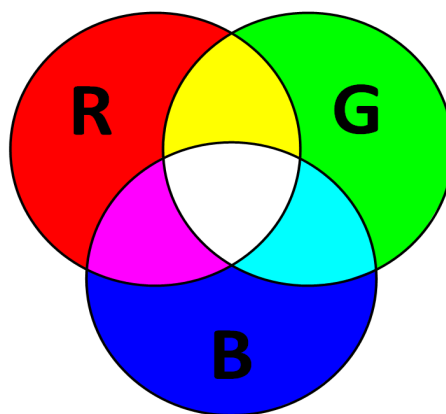


Рисунок 1.23 - Цветовая модель RGB

Каждое из значений можно представить числом, находящимся в диапазоне от 0 до 255, или процентным значением - от 0 до 100%. Если три значения равны 0, то получится чёрный цвет. А если все три цветовых компонента установлены в значение 255, то получится белый цвет. То же самое справедливо и при использовании процентных значений.

Для спецификации цвета в HTML существуют два варианта:

- символьная нотация, например, maroon;
- цифровое обозначение, которое контролирует, каким образом цвет формируется из основных цветов - красного, зеленого и голубого - в так называемое RGB цветовое пространство, при этом обозначение должно быть взято в кавычки (комбинация RGB: “#RRGGBB”). Для задания цифрового обозначения цвета в HTML используются шестнадцатеричная и десятичные формы записи. Например, “#800000” и rgb (128,0,0).

Символьные нотации намного легче и более понятны. С другой стороны, численные обозначения дают больше возможностей. К ним, в частности относится возможность подбора наиболее близких по цветовой гамме цветов при создании единого стиля страницы.

Далее представлены примеры использования этих вариантов.

div {color: purple;}

H1 {color: rgb(255, 0, 255);}

P {color: #800080;}

При описании цветов с использованием цветовой модели RGB можно, помимо цветовых компонентов, указывать и значение, соответствующее альфа-каналу, которое позволяет управлять прозрачностью цвета. Это помогает в работе с насыщенностью цветов, что весьма полезно. Ниже рассматриваются некоторые варианты использования цветов, при описании которых используется альфа-канал. Для работы с такими цветами применяется функция `rgba()`:

```
div { /* Чёрный фон с 50% прозрачностью */ background-color: rgba(0, 0, 0, 0.5); }
```

Если значение, соответствующее альфа-каналу, равно нулю, тогда основной или фоновый цвет элемента окажется полностью прозрачным, то есть - невидимым.

Если пара значений в описании в шестнадцатеричном формате одного компонента цвета идентична, одно из таких значений можно убрать. Например:

```
body {color: #222222; /* Это #222 */
```

```
body {color: #000000; /* Это #000 */
```

Значение `#222222` эквивалентно значению `#222`. Если представить исходный шестнадцатеричный код в виде 22, 22, 22, то, чтобы получить из него код 2, 2, 2, достаточно взять из каждой пары значений по одному.

Наиболее популярные цвета, используемые в HTML, их символьные и числовые эквиваленты представлены в таблице 1.2.

Таблица 1.2 - Таблица HTML цветов

Имя цвета	HEX	RGB	Имя цвета	HEX	RGB
Black	#000000	0, 0, 0	Coral	#FF7F50	255, 127, 80
Gray	#808080	128, 128, 128	Tomato	#FF6347	255, 99, 71
Silver	#C0C0C0	192, 192, 192	OrangeRed	#FF4500	255, 69, 0
White	#FFFFFF	255, 255, 255	DarkOrange	#FF8C00	255, 140, 0
Fuchsia	#FF00FF	255, 0, 255	Gold	#FFD700	255, 215, 0
Purple	#800080	128, 0, 128	LightYellow	#FFFFE0	255, 255, 224
Red	#FF0000	255, 0, 0	LemonChiffon	#FFFACD	255, 250, 205
Maroon	#800000	128, 0, 0	Lavender	#E6E6FA	230, 230, 250

Продолжение таблицы 1.2

Yellow	#FFFF00	255, 255, 0	Thistle	#D8BFD8	216, 191, 216
Olive	#808000	128, 128, 0	Plum	#DDA0DD	221, 160, 221
Lime	#00FF00	0, 255, 0	Indigo	#4B0082	75, 0, 130
Green	#008000	0, 128, 0	SlateBlue	#6A5ACD	106, 90, 205
Aqua	#00FFFF	0, 255, 255	Goldenrod	#DAA520	218, 165, 32
Teal	#008080	0, 128, 128	DarkGoldenRod	#B8860B	184, 134, 11
Blue	#0000FF	0, 0, 255	Peru	#CD853F	205, 133, 63
Navy	#000080	0, 0, 128	Chocolate	#D2691E	210, 105, 30
FireBrick	#B22222	178, 34, 34	SpringGreen	#00FF7F	0, 255, 127
DarkRed	#8B0000	139, 0, 0	MediumSeaGreen	#3CB371	60, 179, 113
Salmon	#FA8072	250, 128, 114	SeaGreen	#2E8B57	46, 139, 87
DeepPink	#FF1493	255, 20, 147	ForestGreen	#228B22	34, 139, 34
Aquamarine	#7FFFD4	127, 255, 212	MediumAquaMarine	#66CDAA	102, 205, 170

Далее приведен пример использования цветов при создании HTML страниц.

```
<body style="background-color: gainsboro">
```

```
<h2>Использование цвета в HTML</h2>
```

```
<h1 style="background-color: cyan">Имя Цвета</h1>
```

```
<h3 style="background-color: #F067FF;">HEX</h3>
```

```
<h3 style="background-color: #F0;">Сокращение HEX</h3>
```

```
<h3 style="background-color: rgb(255, 124, 30)">RGB</h3>
```

```
<h3 style="background-color: rgba(255, 0, 0, 0.5);">RGBA</h3> </body>
```

Реализация данного кода представлена на рисунке 1.24.

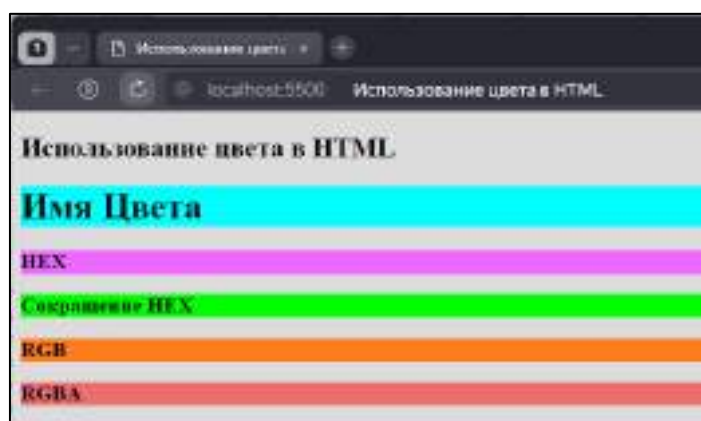


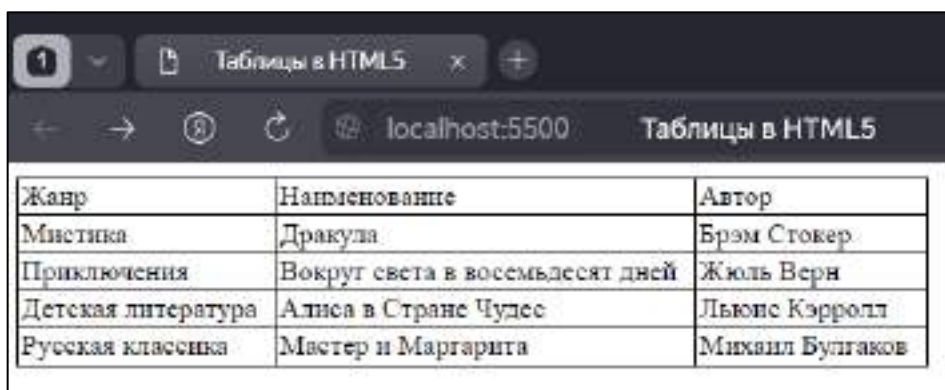
Рисунок 1.24 - Работа с цветами в HTML

1.14 Таблицы в HTML5

Для создания таблиц в html используется элемент table. Каждая таблица между тегами <table> и </table> содержит строки, который представлены элементом tr. А каждая строка между тегами <tr> и </tr> содержит ячейки в виде элементов td. Далее представлен пример простой таблицы.

```
<body><table border=2px><tr>
  <td>Жанр</td> <td>Наименование</td> <td>Автор</td></tr>
<tr> <td>Мистика</td> <td>Дракула</td> <td>Брэм Стокер</td> </tr>
<tr> <td>Приключения</td> <td>Вокруг света в восемьдесят дней</td> <td>Жюль Верн</td>
</tr> <tr> <td>Детская литература</td> <td>Алиса в Стране Чудес</td> <td>Льюис
Кэрролл</td> </tr>
<tr> <td>Русская классика</td> <td>Мастер и Маргарита</td> <td>Михаил
Булгаков</td> </tr></table></body>
```

Реализация данного кода представлена на рисунке 1.25.



Жанр	Наименование	Автор
Мистика	Дракула	Брэм Стокер
Приключения	Вокруг света в восемьдесят дней	Жюль Верн
Детская литература	Алиса в Стране Чудес	Льюис Кэрролл
Русская классика	Мастер и Маргарита	Михаил Булгаков

Рисунок 1.25 - Таблицы в HTML5

Здесь представлено в таблице пять строк, каждая строка имеет по три столбца. При этом первая строка выполняет роль заголовка, а остальные три строки являются содержимым таблицы. Разделения заголовков, футера и тела таблицы в html предусмотрены соответственно элементы thead, tfoot и tbody. Для их применения таблица была изменена следующим образом.

```
<body><table border=2px> <caption><b>Популярные книги</b></caption>
<thead><tr><th>Жанр</th><th>Наименование</th><th>Автор</th></tr></thead>
```



```

<tbody><tr> <td>Мистика</td><td>Дракула</td><td>Брэм Стокер</td></tr> <tr>
<td>Приключения</td><td>Вокруг света в восемьдесят дней</td><td>Жюль Верн</td></tr>
<tr><td>Детская литература</td><td>Алиса в Стране Чудес</td><td>Льюис Кэрролл</td>
</tr> <tr> <td>Русская классика</td><td>Мастер и Маргарита</td><td>Михаил
Булгаков</td></tr> </tbody><tfoot>

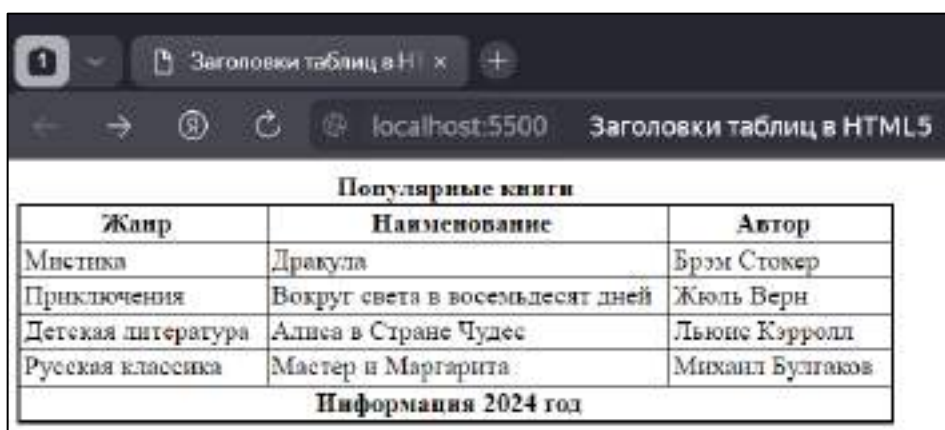
```

```

<tr> <th colspan="3">Информация 2024 год</th></tr> </tfoot></table></table>

```

В элемент `thead` заключается строка заголовков. Для ячеек заголовков используется не элемент `td`, а `th`. Элемент `th` выделяет заголовок жирным. А все остальные строки заключаются в `tbody`. Элемент `tfoot` определяет подвал таблицы или футер. Здесь обычно выводится некоторая вспомогательная информация по отношению к таблице. Реализация данного кода представлена на рисунке 1.26.



Популярные книги		
Жанр	Наименование	Автор
Мистика	Дракула	Брэм Стокер
Приключения	Вокруг света в восемьдесят дней	Жюль Верн
Детская литература	Алиса в Стране Чудес	Льюис Кэрролл
Русская классика	Мастер и Маргарита	Михаил Булгаков
Информация 2024 год		

Рисунок 1.26 - Заголовки таблиц в HTML5

Кроме заголовков столбцов с помощью элемента `caption` можно задать общий заголовок для таблицы. Для объединения столбцов таблицы используется атрибута `colspan`, который указывает на какое количество столбцов раздвигается данная ячейка. В данном примере футер таблицы содержит только один столбец, который раздвигается по ширине трех столбцов с помощью атрибута `colspan="3"`.

Также с помощью атрибута `rowspan` можно раздвигать ячейку на определенное количество строк. Далее представлен пример объединения ячеек по столбцам и строкам.

```

<style> td { width: 80px; height:80px;
border: solid 2px silver;
text-align: center; } </style> </head>

```

```

<body> <h2>Мультфильмы для детей</h2> <table>
  <tr> <td colspan="2" style="background-color: #bb84e8;"> Лунтик. Возвращение
домой</td> <td> Гадкий Я 4</td> </tr>
  <tr> <td> Головоломка 2</td> <td rowspan="2" style="background-color: #00ffff;">
Большое путешествие. Вокруг света</td> <td> 10 жизней </td> </tr>
  <tr> <td> Пушистый вояж </td> <td> Кунг-фу Панда 4</td> </tr> </table>
</body>

```

Реализация данного кода представлена на рисунке 1.27.

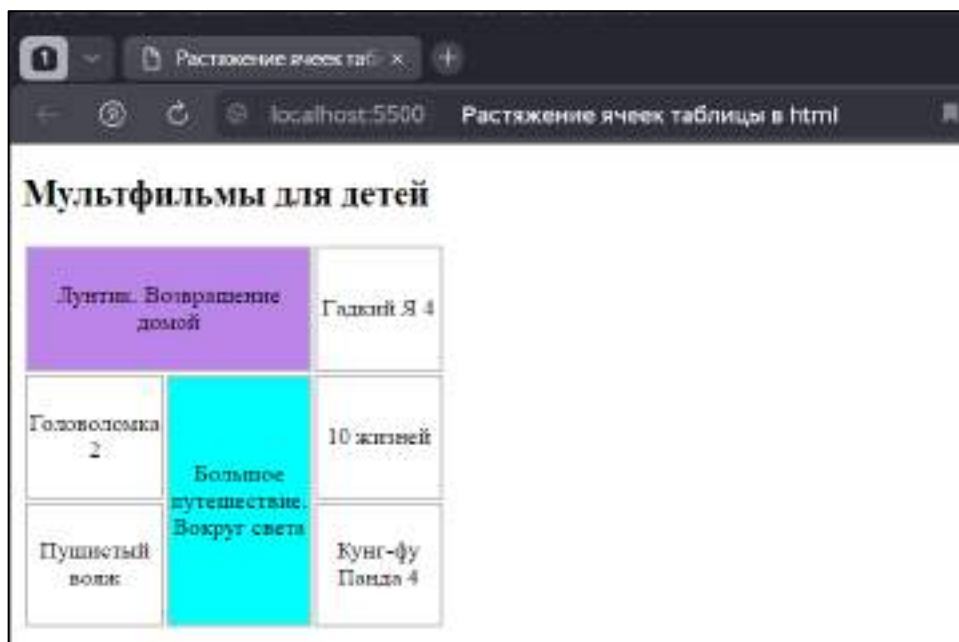


Рисунок 1.27 - Растяжение ячеек таблицы в html

1.15 Гипертекстовые ссылки в HTML

Ссылки, которые представлены элементом `<a>`, играют важную роль - они обеспечивают навигацию между отдельными документами. Основным атрибутом тега является атрибут `href`, в котором задается определяет адрес ссылки.

Тег `a(anchor)` делится на видимую часть, то есть то что отображается на экране и скрытую часть.

```
<a href=index.html> Ссылка</a>
```

Тег <a> помимо атрибута href может иметь следующие:

- hreflang: указывает на язык документа, на который ведет данная ссылка;
- media: определяет устройство, для которого предназначена ссылка;
- rel: определяет отношение между данным документом и ресурсом, на который ведет ссылка;
- target: определяет, как документ по ссылке должен открываться;
- type: указывает на mime-тип ресурса по ссылке.

Наиболее важным из перечисленных является атрибут href. В зависимости от того, какой указан адрес ресурса в атрибуте href ссылки разделяют на абсолютные, относительные и внутренние. Далее представлен пример абсолютной ссылки, в которой в качестве адреса ресурса указан www.osu.ru (рисунок 1.28).

```
<body>  
<a href="http://www.osu.ru/">  
Сайт Оренбургского государственного университета</a>  
</body>
```

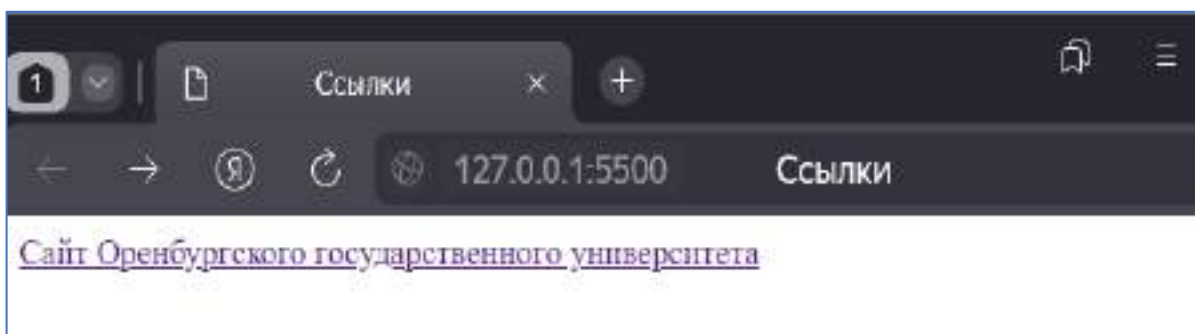


Рисунок 1.28 - Ссылки в HTML5

В следующем примере представлен пример использования относительных ссылок, где в качестве адреса ресурса используется относительный путь до файлов glava1.html, glava2.html, glava3.html, которые находятся в той же папке.

```
<body> <h2> Содержание курса по Веб-программированию</h2>  
<a href="glava1.html">Глава 1. Основы HTML.</a>  
<a href="glava2.html">Глава 2. Введение в JavaScript.</a>  
<a href="glava3.html">Глава 3. Каскадные таблицы стилей CSS. </a> </body>
```

Реализация данного кода представлена на рисунке 1.29.

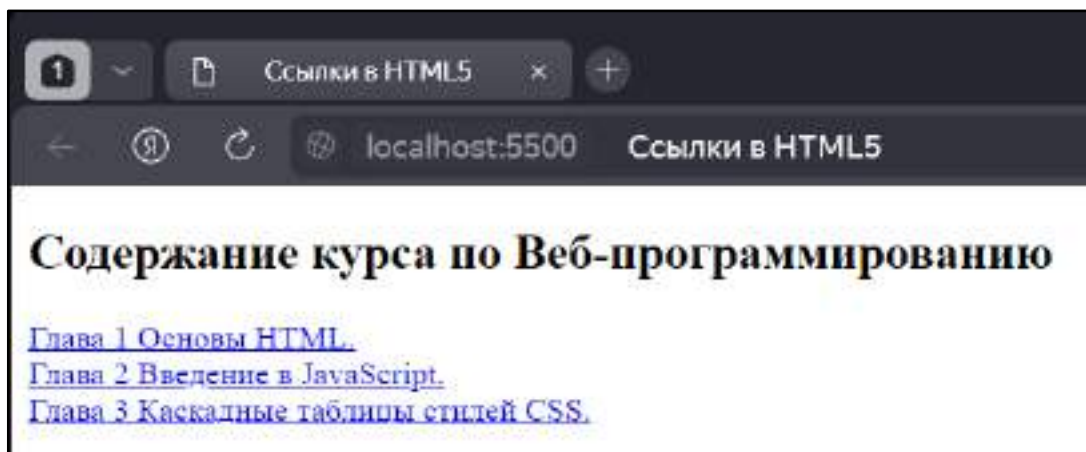


Рисунок 1.29 - Ссылки в HTML5

1.15.1 Навигация внутри документа

С помощью тега `<a>` также можно задать внутренние ссылки, которые будут переходить к определенным блокам внутри элементов.

```
<body> <a href="#paragraph1">Глава 1</a> | <a href="#paragraph2">Глава 2</a> |  
<a href="#paragraph3">Глава 3</a>  
  <h2 id="paragraph1"> Глава 1</h2><p>Содержание главы 1</p>  
  <h2 id="paragraph2"> Глава 2</h2> <p>Содержание главы 2</p>  
  <h2 id="paragraph3"> Глава 3</h2><p>Содержание главы 3</p>  
</body>
```

Реализация данного кода представлена на рисунке 1.30.

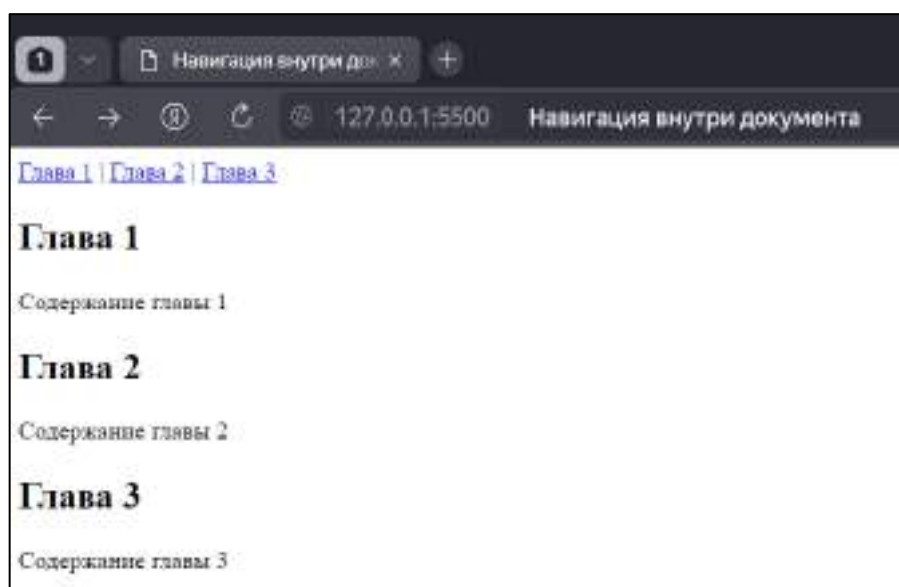


Рисунок 1.30 - Навигация внутри документа

Чтобы определить внутреннюю ссылку, указывается знак решетки (#), после которого идет id того элемента, к которому надо осуществить переход. В данном случае переход будет идти к заголовкам h2.

1.15.2 Атрибут target

По умолчанию ресурсы, на которые ведут ссылки, открываются в том же окне. С помощью атрибута target можно переопределить это действие. Атрибут target может принимать следующие значения:

- `_blank`: открытие html-документа в новом окне или вкладке браузера;
- `_self`: открытие html-документа в том же фрейме (или окне);
- `_parent`: открытие документа в родительском фрейме, если ссылка расположена во внутреннем фрейме;
- `_top`: открытие html-документа на все окно браузера;
- `framename`: открытие html-документа во фрейме, который называется `framename`.

Например, открытие документа по ссылке в новом окне:

```
<a href="http://osu.com/web/html5/" target="_blank">Учебник по HTML5</a>
```

1.15.3 Стилизация ссылок

По умолчанию ссылка уже имеет некоторый цвет (один из оттенков синего), кроме того, она имеет подчеркивание. При нажатии на ссылку она становится активной и приобретает красный цвет, а после перехода по ссылке эта ссылка может окраситься в другой цвет (как правило, в фиолетовый). Подобная стилизация задается многими браузерами по умолчанию, но ее можно переопределить. Определяются стили для ссылок в различных состояниях:

a: link применяется для ссылок в обычном состоянии, когда они не нажаты и на них не наведен указатель мыши,

a: visited - указывает на состояние ссылки, по которой уже был осуществлен переход,

a: hover - указывает на состояние ссылки, на которую навели указатель мыши,

a: active - указывает на ссылку в нажатом состоянии,

Далее представлен пример использования всех видов ссылок.

```
<head> <style>
    a:link {color:#5fffaa; text-decoration:none};
    a:visited {color:#33c1ff; text-decoration:none};
    a:hover {color:#020096; text-decoration:underline};
    a:active {color:#aqua; text-decoration:underline}; </style> </head>
<body> <h1>Фильмы для всей семьи</h1>
<a href="index.html"> Сто лет тому вперед </a>
<a href="index.html">Чебурашка</a>
<a href="index.html">По щучьему веленью</a>
<a href="index.html">Последний богатырь</a> </body>
```

Реализация данного кода представлена на рисунке 1.31.

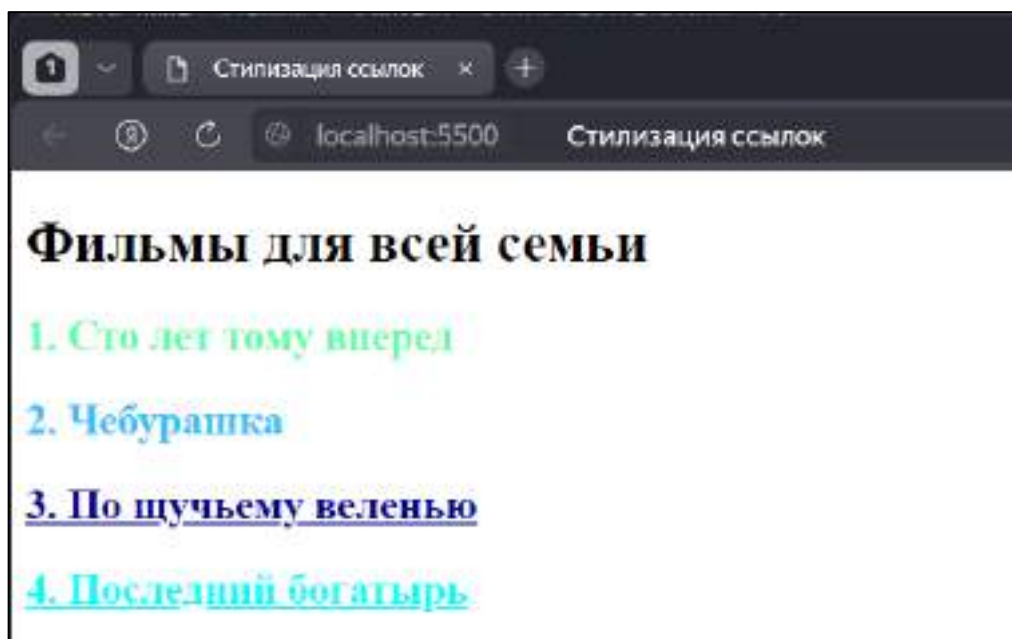


Рисунок 1.31 - Стилизация ссылок

Например, для стиля color: aqua устанавливает цвет для активной ссылки. А стиль text-decoration устанавливает подчеркивание: если значение underline, то ссылка подчеркнута, если none, то подчеркивание отсутствует.

1.15.4 Ссылка-картинка

Поместив внутрь элемента `<a>` элемент ``, можно сделать ссылку-изображение.

```
<a href="aboutcat.html">  
 </a>
```

Реализация данного кода представлена на рисунке 1.32.

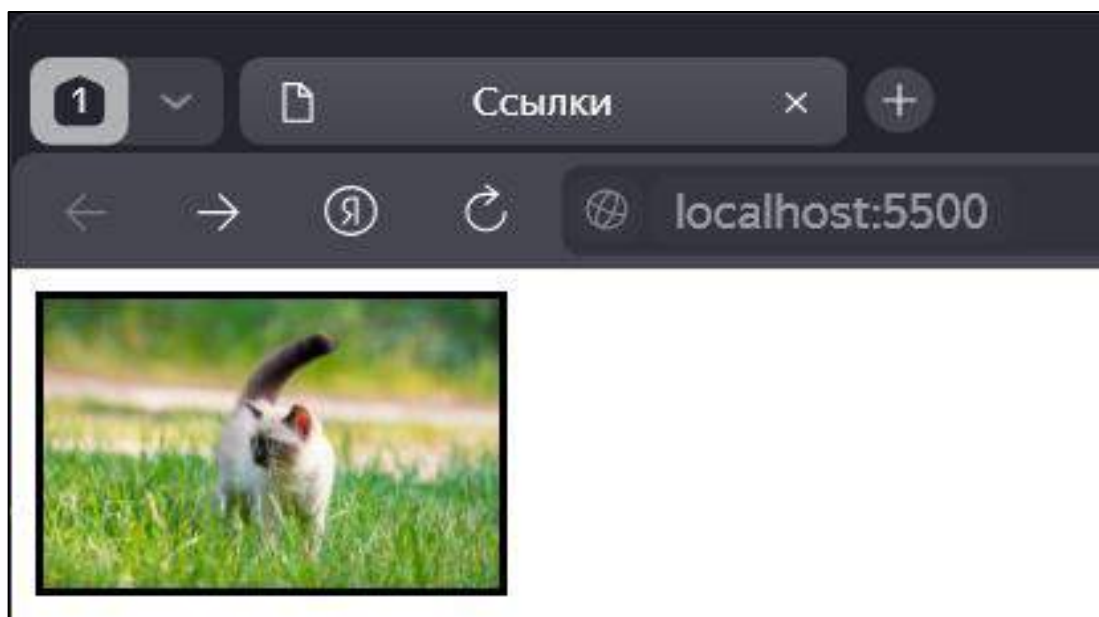


Рисунок 1.32 – Рисунок-ссылка

В данном примере при нажатии на рисунок-ссылку будет реализован переход на страницу `aboutcat.html`, в которой описывается информация о кошках.

1.16 Элементы `figure` и `figcaption`

Элемент `figure` применяется для аннотирования различных иллюстраций, диаграмм, фотографий и т.д. А элемент `figcaption` включает подпись к содержимому внутри элемента `figure`.

Для использования элемента `figure` нам надо поместить в него некоторое содержимое, например, изображение.

```
<body> <div>
<p>Летний пейзаж ... </p>
  <figure>
<figcaption>Лето 2024</figcaption>

  </figure>
<p>Летний пейзаж ... </p> </div>
</body>
```

Реализация данного кода представлена на рисунке 1.33.

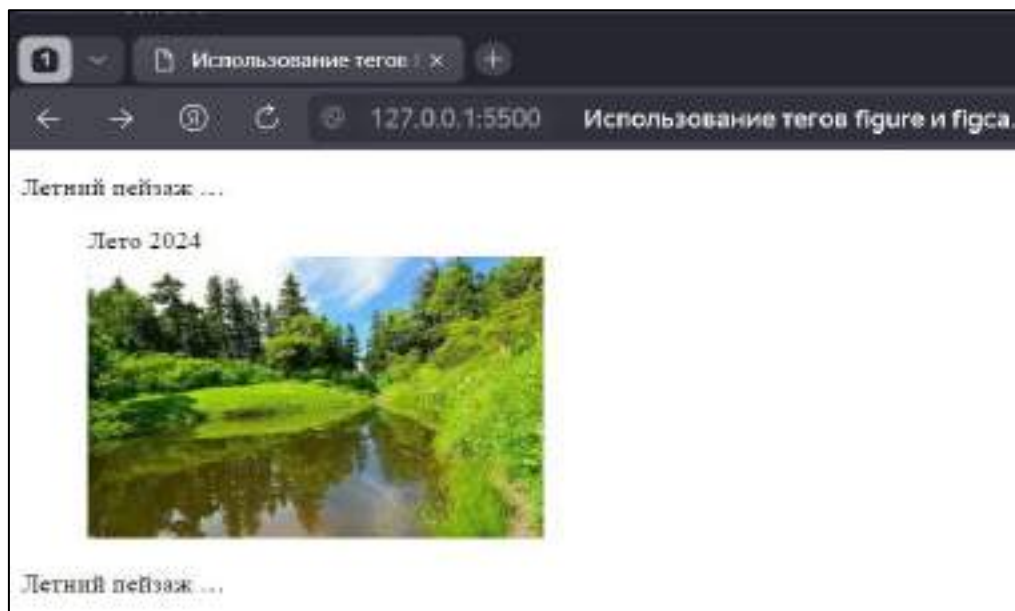


Рисунок 1.33 - Использование тегов `figure` и `figcaption` в HTML5

1.17 Использование плавающих фреймов в HTML

Плавающие фреймы позволяют встраивать на веб-страницу еще какую-нибудь другую веб-страницу. Фреймы представлены элементом `iframe`. Элемент `iframe` не содержит в себе никакого содержимого. Вся его настройка производится с помощью атрибутов:

- src: устанавливает полный путь к загружаемому ресурсу;
- width: ширина фрейма;
- height: высота фрейма.

Например, с помощью использования плавающего фрейма можно встроить на страницу карту размером 560 на 400 пикселей.

```
<body> <iframe id="inlineFrameExample"
  title="Inline Frame Map"
  width="560"
  height="400"
  frameborder="1"
  allowfullscreen="true"
  src="https://yandex.ru/map-widget/v1/-/CBFkaYSE0A"> </iframe> </body>
```

Реализация данного кода представлена на рисунке 1.34.

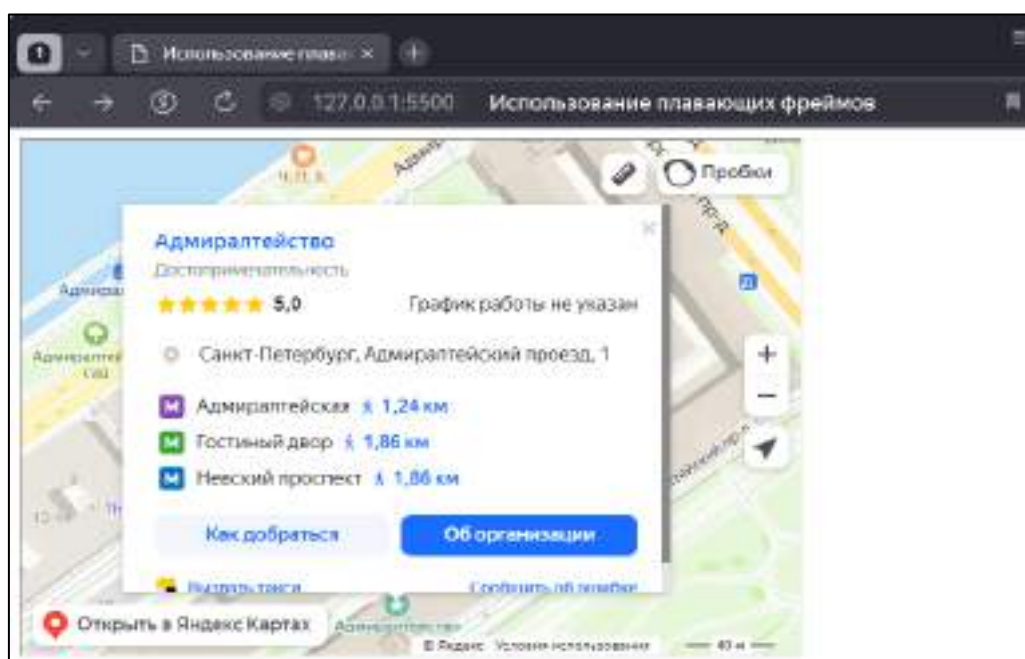


Рисунок 1.34 – Использование плавающих фреймов

До недавнего времени помимо плавающих фреймов (iframe) использовались фреймы (frames), которые разбивали окно браузера на несколько областей, в каждую из которых загружался свой документ. Однако, на сегодняшний день использование фреймов не рекомендовано из-за сильной перегрузки трафика, а также затруднения работы поисковых машин.

1.18 Контрольные вопросы

- 1) Рассказать о работе с метаданными веб-документа. Назначения тега META.
- 2) Перечислить теги форматирования текста в HTML.
- 3) Работа с мультимедиа. Описать теги работы с изображениями и музыкой.
- 4) Описать работу с ссылками в HTML. Относительные и абсолютные ссылки.
- 5) Формат RGB. Рассказать об обозначении цветов в HTML.

1.19 Тестирование по главе

- 1) Элемент ... указывает базовый адрес
 - а) base;
 - б) head;
 - в) meta;
 - г) title.

- 2) Элемент ... структурирует контент на сайте, группирует содержимое в блоки
 - а) p;
 - б) div;
 - в) pre;
 - г) span.

- 3) Самый крупный заголовок имеет тег ...
 - а) h1;
 - б) h2;
 - в) h4;
 - г) h6.

4) Тег ... выделяет текст курсивом

- а) ;
- б) ;
- в) <i>;
- г) <s>.

5) Атрибут ... используется для задания пути к изображению, а ... - для описания изображения

- а) src, alt;
- б) alt, src;
- в) img, alt;
- г) src, img.

6) Для нумерованного списка используется элемент ...

- а) ul;
- б) ol;
- в) demical;
- г) li.

7) Элемент ... создает раскрываемый блок

- а) li;
- б) summary;
- в) details;
- г) head.

8) Строки в таблицах обозначаются элементом...

- а) tr;
- б) td;
- в) ts;
- г) dt.

9) Какой атрибут ссылок `<a>` указывает адрес ресурса?

- а) rel;
- б) media;
- в) href;
- г) target.

10) Выберите верное сокращение цвета.

- а) #FF00FF - #F0F;
- б) #DDA0DD - #DA0D;
- в) #D8BFD8 - #DBD;
- г) #E6E6FA - #E6FA.

2 Работа с формами в HTML5

2.1 Задание формы

Формы в HTML представляют один из способов для ввода и отправки данных. Они являются наиболее популярным способом “обратной связи” с пользователем. С помощью HTML можно создавать как простые формы, предполагающие выбор одного из нескольких ответов, так и сложные формы для заказов или для того, чтобы получить от пользователей страницы какие-либо комментарии и пожелания.

Пользователь может взаимодействовать с веб-сервером разными способами: регистрация и вход на сайтах, ввод личной информации, фильтрация контента, поиск, загрузка файлов. С помощью форм можно предоставить пользователям возможность взаимодействовать с сайтом или приложением, проверять введенную информацию и отправлять данные на сервер [1]. HTML имеет несколько интерактивных элементов управления формами: текстовые поля, переключатели, флажки, выпадающие списки, кнопки отправки. Для реализации этих элементов в HTML используется тег `<FORM>`.

`<form method=“get|post” action=“url” enctype> Элементы формы и другие элементы HTML</form>`

Элемент `<FORM>` обозначает документ как форму и определяет границы использования других тегов, размещаемых в форме:

- `method`: устанавливает метод отправки данных на сервер. Допустимы два значения: `post` и `get`;
- значение `post` позволяет передать данные на веб-сервер через специальные заголовки. А значение `get` позволяет передать данные через строку запроса;
- `action`: устанавливает адрес, на который передаются данные формы;
- `enctype`: устанавливает тип передаваемых данных.

`Enctype` в свою очередь может принимать следующие значения:

- application/x-www-form-urlencoded: кодировка отправляемых данных по умолчанию;
- multipart/form-data: эта кодировка применяется при отправке файлов;
- text/plain: эта кодировка применяется при отправке простой текстовой информации.

Далее представлен пример создания простейшей формы.

```
<body> <form method="post" action="http://localhost:5000/login.js">  
    <input name="regist"/>  
    <input type="submit" value="Войти" />  
</form> </body>
```

Реализация данного кода представлена на рисунке 2.1.

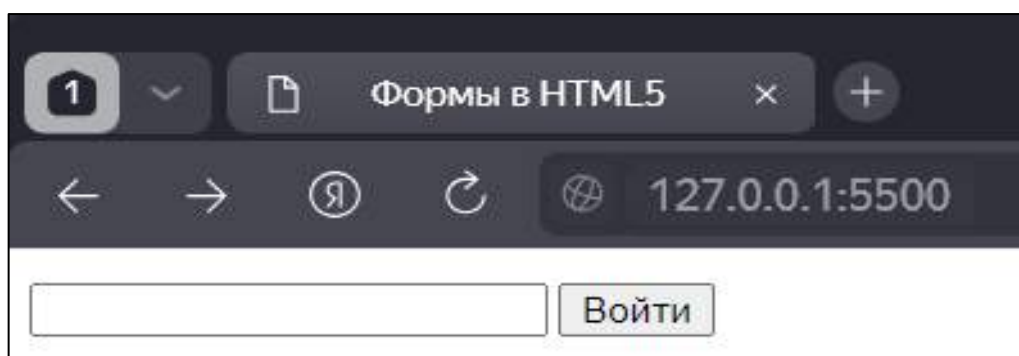


Рисунок 2.1 - Формы в HTML5

В данном примере использовался метод post для отправки данных из формы на сервер. Все значения формы отправляются в теле запроса, а адресом служит строка `http://localhost:5000/login.js`. Отправка данного запроса приведена на рисунке 2.2.

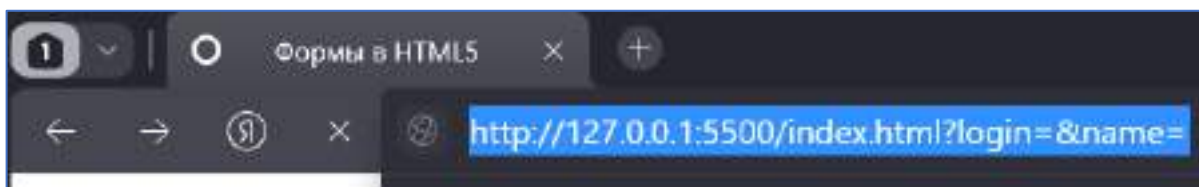


Рисунок 2.2 - Отправка запроса

Как правило, по указанному адресу работает веб-сервер, который, используя одну из технологий серверной стороны (PHP, NodeJS, ASP.NET и т.д.), он может получать запросы и возвращать ответ.

Часто веб-браузеры запоминают вводимые данные, и при вводе могут выдавать список подсказок из ранее введенных слов (рисунок 2.3).

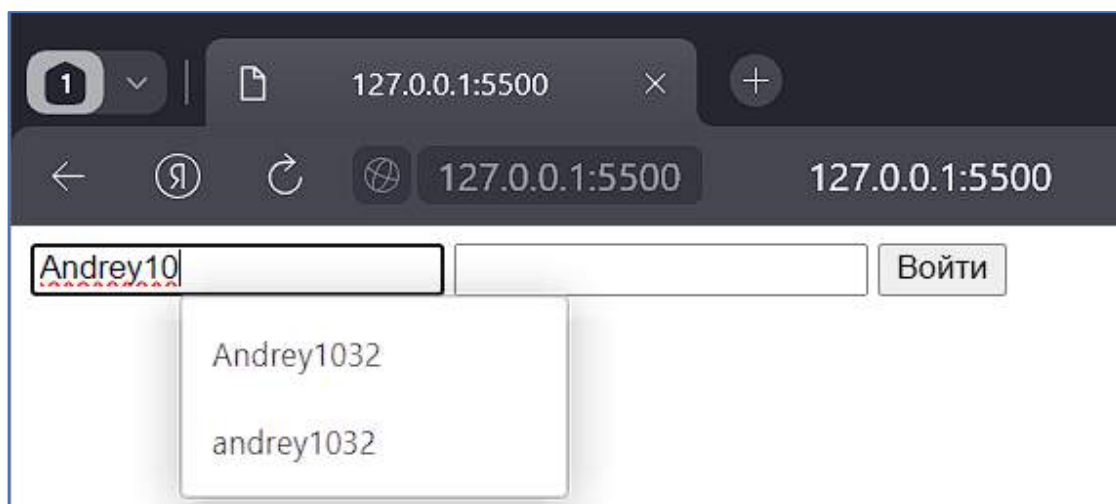


Рисунок 2.3 - Autocomplete в HTML5

С помощью атрибута `autocomplete` можно отключить авто дополнение:

```
<form method="post" autocomplete="off" action="http://localhost:5000/login.js">  
  <input name="regist" /><input name="password" />  
  <input type="submit" value="Войти" /> </form>
```

Если необходимо включить автодополнение только для каких-то определенных полей, то можно применить к ним атрибут `autocomplete="on"`:

```
<form method="post" autocomplete="off" action="http://localhost:5000/login.js">  
  <input name="regist" />  
  <input name="password" autocomplete="on" />  
  <input type="submit" value="Войти"/>  
</form>
```

Теперь для всей формы, кроме второго поля, будет отключено автодополнение.

2.2 Элементы форм

Формы состоят из определенного количества элементов ввода. Все элементы ввода помещаются между тегами `<form>` и `</form>`.

Наиболее распространенным элементом ввода является элемент `input`. Однако реальное действие этого элемента зависит от того, какое значение установлено у его атрибута `type`. А он может принимать следующие значения:

- `text`: обычное текстовое поле;
- `password`: тоже текстовое поле, только вместо вводимых символов отображаются звездочки, поэтому в основном используется для ввода пароля;
- `radio`: радиокнопка или переключатель. Из группы радиокнопок можно выбрать только одну;
- `checkbox`: элемент флажок, который может находиться в отмеченном или неотмеченном состоянии;
- `hidden`: скрытое поле;
- `submit`: кнопка отправки формы;
- `color`: поле для ввода цвета;
- `date`: поле для ввода даты;
- `datetime`: поле для ввода даты и времени с учетом часового пояса;
- `datetime-local`: поле для ввода даты и времени без учета часового пояса;
- `email`: поле для ввода адреса электронной почты;
- `month`: поле для ввода года и месяца;
- `number`: поле для ввода чисел;
- `range`: ползунок для выбора числа из некоторого диапазона;
- `tel`: поле для ввода телефона;
- `time`: поле для ввода времени;
- `week`: поле для ввода года и недели;
- `url`: поле для ввода адреса `url`;
- `file`: поле для выбора отправляемого файла;
- `image`: создает кнопку в виде картинки.

Кроме элемента `input` при задании элементов форм можно использовать теги:

- `button`: кнопка;
- `select`: выпадающий список;
- `label`: метка, отображаемая рядом с полем ввода;

–textarea: многострочное текстовое поле.

У всех элементов ввода можно установить атрибуты name и value. Эти атрибуты имеют важное значение.

По атрибуту name можно идентифицировать поле ввода, а атрибут value позволяет установить значение поля ввода. Например:

```
<body> <form method="get" action="index.html">  
    <input type="text" name="login" value="Елена"/>  
    <input type="password" name="password"/>  
    <input type="submit" value="Вход" />  
</form> </body>
```

Здесь текстовое поле имеет значение “Елена” (как указано в атрибуте value), поэтому при загрузке веб-страницы в этом поле можно увидеть данный текст (рисунок 2.4).

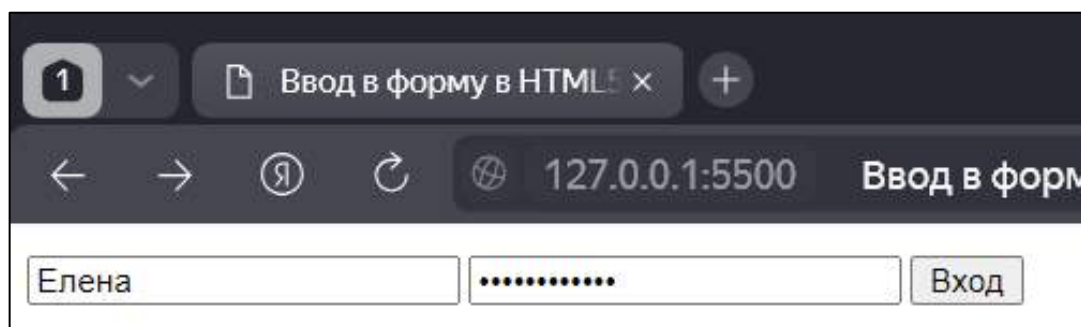


Рисунок 2.4 - Ввод в форму в HTML5

Поскольку методом отправки данных формы является метод “get”, то данные будут отправляться через строку запроса. Так как в данном случае не важно, как данные будут приниматься, не важен сервер, который получает данные, поэтому в качестве адреса установлена та же самая страница - то есть файл index.html. При отправке можно увидеть введенные данные в строке запроса (рисунок 2.5).

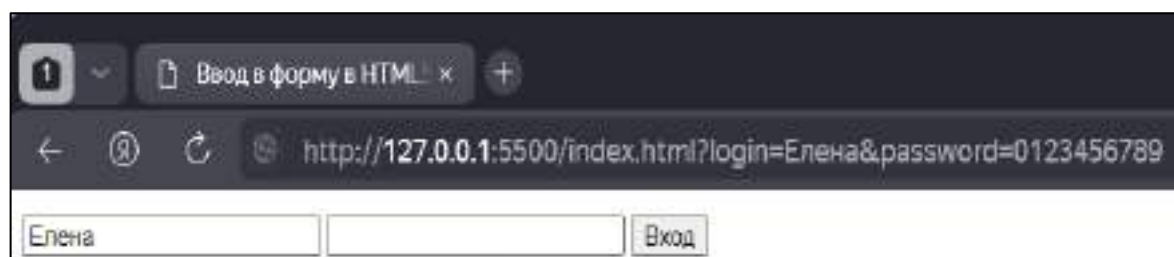


Рисунок 2.5 - Ввод в форму в HTML5

В строке запроса нас интересует следующий кусочек:

login=Елена&password=0123456789

При отправке формы браузер соединяет все данные в набор пар “ключ-значение”. В этом случае две таких пары: login=Елена и password=0123456789. Ключом в этих парах выступает название поля ввода, которое определяется атрибутом name, а значением - собственно то значение, которое введено в поле ввода (или значение атрибута value).

Получив эти данные, сервер легко может узнать, какие значения в какие поля ввода были введены пользователем.

2.3 Кнопки

Кнопки в HTML5 представлены элементом button. В зависимости от значения атрибута type можно создать различные типы кнопок:

- submit: кнопка, используемая для отправки формы;
- reset: кнопка сброса значений формы;
- button: кнопка без какого-либо специального назначения.

Например,

<button type="submit" value="Отправить">

< button type="reset" value="Отмена">

Если кнопка используется для отправки формы, то есть у нее установлен атрибут type=“submit”, то можно задать у нее ряд дополнительных атрибутов:

- form: определяет форму, за которой закреплена кнопка отправки;
- formaction: устанавливает адрес, на который отправляется форма. Если у элемента form задан атрибут action, то он переопределяется;
- formenctype: устанавливает формат отправки данных. Если у элемента form установлен атрибут enctype, то он переопределяется;

-formmethod: устанавливает метод отправки формы (post или get). Если у элемента form установлен атрибут method, то он переопределяется.

Далее представлен пример, использования тега button для создания кнопок отправки и сброса при передаче данных об имени и фамилии пользователя.

```
<body><form>
<h2>Введите свои данные </h2>
<p><input type="text" name="Imya"/></p>
<p><input type="text" name="Famil"/></p>
<p>
        <button                type="submit"                formmethod="get"
formaction="index.html">Отправить</button>
        <button type="reset">Отмена</button> </p></form></body>
```

На рисунке 2.6 представлен пример реализации данного кода.

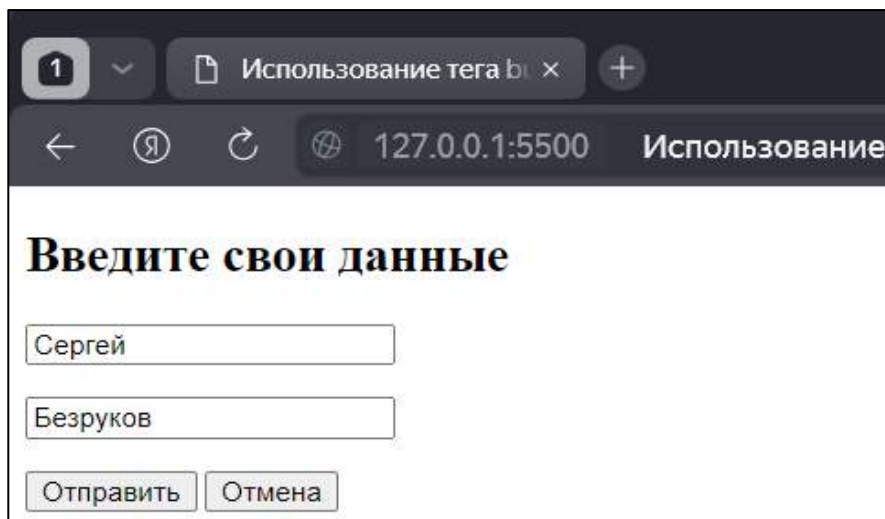


Рисунок 2.6 - Использование тега button для создания кнопок

Кроме элемента button для создания кнопок можно использовать элемент input, у которого атрибут равен submit или reset.

```
<input type="submit" value="Отправить">
```

```
<input type="reset" value="Отмена">
```

Если необходимо использовать в качестве кнопки изображение, то можно использовать элемент input с атрибутом type="image".

```
<body> <form>
```

```
<p><input type="text" name="search" />
```

```
<input type="image" src="search.png" name="submit" /></p>
</form> </body>
```

Реализация данного кода представлена на рисунке 2.7.

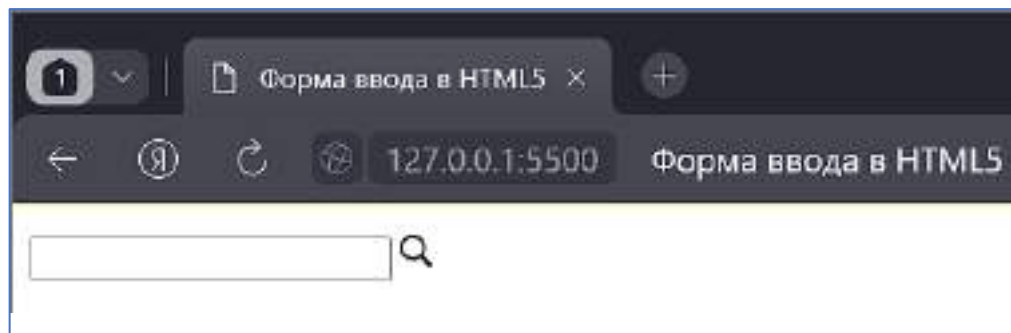


Рисунок 2.7 - Кнопка-изображение в HTML5

Кроме наличия изображения в остальном эта кнопка будет аналогична стандартной кнопке отправки `input type="submit"` или `button type="submit"`.

2.4 Текстовые поля

Однострочное текстовое поле создается с помощью элемента `input`, когда его атрибут `type` имеет значение `text`.

```
<input type="text" name="login" />
```

Текстовое поле используя несколько дополнительных атрибутов:

- `dir`: направление текста;
- `list`: список подсказок для ввода в поле;
- `maxlength`: максимальное количество символов;
- `pattern`: шаблон, которому должен соответствовать введенный текст;
- `placeholder`: текст, который будет отображаться по умолчанию в текстовом поле;
- `readonly`: делает текстовое поле доступным только для чтения;
- `required`: указывает, что текстовое поле должно иметь значение;

- size: определяет ширину текстового поля в видимых символах;
- value: устанавливает значение по умолчанию в текстовом поле [11].

Далее представлен пример применения некоторых атрибутов при создании текстового поля.

```
<body> <form>
<p><input type="text" name="userName" placeholder="ФИО" size="17" /></p>
<p> <input type="text" name="userPhone" placeholder="Почта" size="17"
maxlength="12" /> </p>
<p> <button type="submit">Готово</button> <button type="reset">Отменить</button>
</p> </form>
</body>
```

Реализация данного кода представлена на рисунке 2.8.

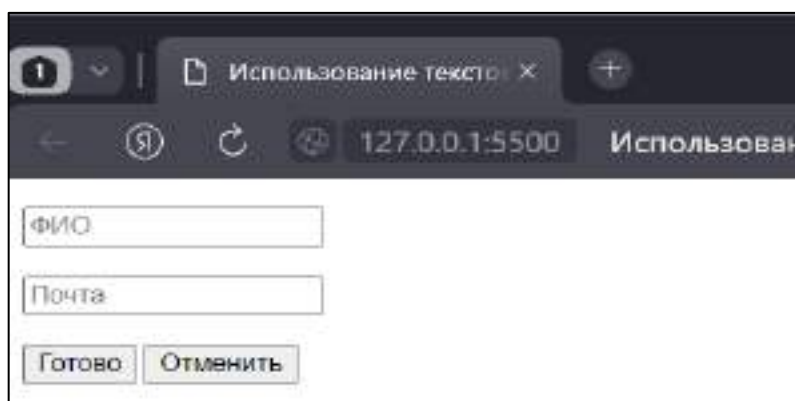


Рисунок 2.8 - Текстовые поля в HTML5

В этом примере атрибуты maxlength и size сразу устанавливаются во втором текстовом поле. В данном случае размер - т.е. количество символов, уместяющихся в видимом пространстве поля, больше допустимого количества символов. Однако нельзя ввести длину символов, превышающую максимальную.

В этом случае также важно различать атрибуты value и placeholder, хотя оба они устанавливают видимый текст в поле. Placeholder устанавливает подсказку, он неактивен и выделяется серым цветом. Значение value представляет собой текст по умолчанию, введенный в поле.

```
<p><input type="text" name="userName" value="Иван" /></p>
<p><input type="text" name="userPhone" placeholder="Почта" /></p>
```

В данном случае используются два атрибута placeholder обозначения имени и почты (рисунок 2.9).

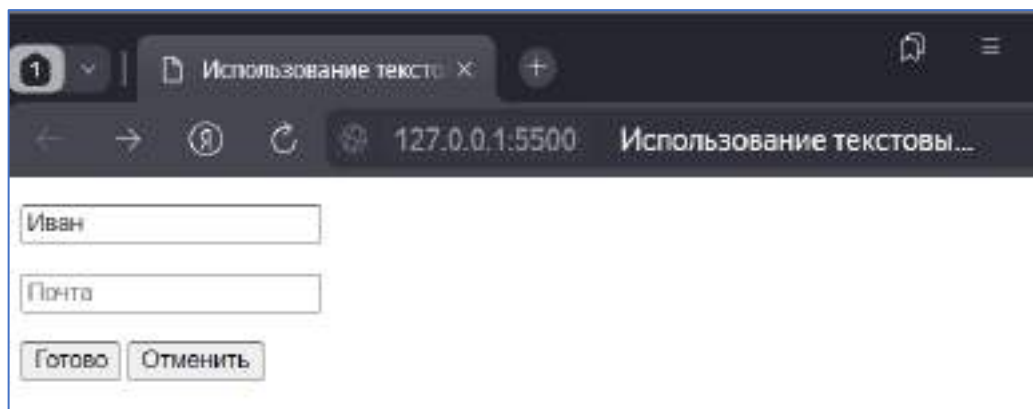


Рисунок 2.9 – Использование placeholder в HTML5

Атрибуты readonly и disabled делают текстовое поле недоступным, однако сопровождаются разным визуальным эффектом. В случае с disabled текстовое поле затемняется (рисунок 2.10).

```
<p><input type="text" name="userName" value="Иван" readonly /> </p>
```

```
<p><input type="text" name="userPhone" value="ivan@mail.ru" disabled /></p>
```

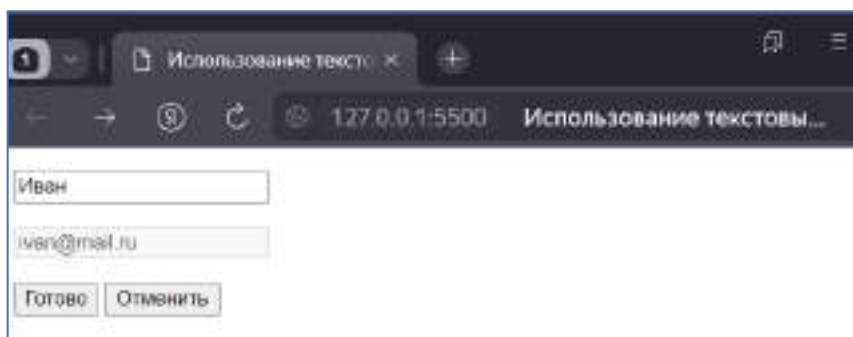


Рисунок 2.10 - Disabled и readonly в HTML5

Среди атрибутов текстового поля также стоит отметить такой атрибут как list. Атрибут содержит ссылку на элемент datalist, который определяет набор значений, которые появляются в виде подсказок при вводе в текстовое поле. Например:

```
<body>
```

```
<h2>Покупка техники</h2>
```

```
<form><input list="nameList" type="text" name="model" placeholder="Наименование  
техники" /></p> <p> <button type="submit">Готово</button> </p> </form>
```

```

<datalist id="nameList">
  <option value="Телефон Samsung Galaxy A25" label="20000 руб."/>
  <option value="Ноутбук Acer Swift Go 16 ">100000 руб.</option>
  <option value="Принтер HP375"/>30000 руб.</option>
</datalist>
</body>

```

Пример реализации данного кода представлен на рисунке 2.11.

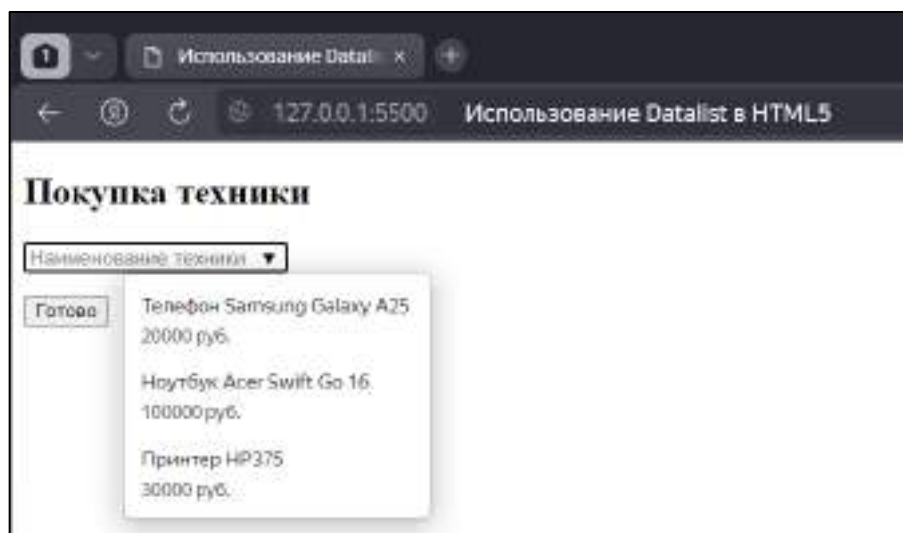


Рисунок 2.11 – Использование Datalist в HTML5

Атрибут list текстового поля указывает на id элемента datalist. Сам элемент datalist определяет элементы списка, используя вложенные элементы option. И когда пользователь вводит текстовое поле, этот список появляется в виде всплывающей подсказки. Для изменения направления текста в теге input используется атрибут dir, который может принимать два значения: ltr (слева направо) и rtl (справа налево).

```

<input type="text" name="username" dir="rtl" />

```

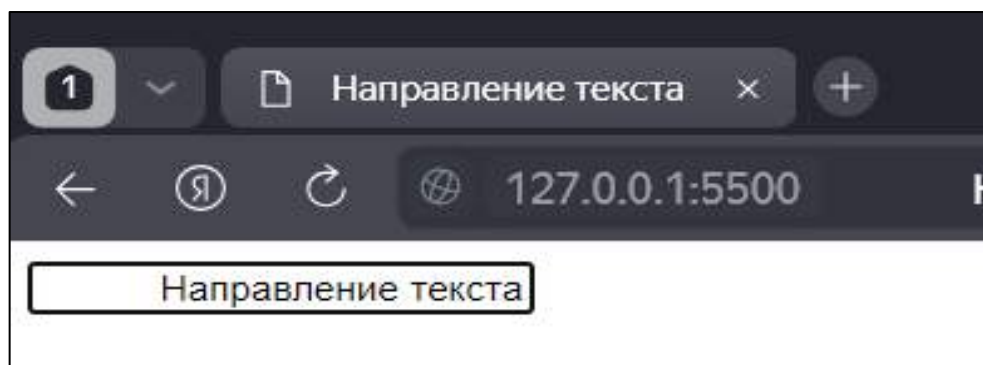


Рисунок 2.12 – Направление текста

2.5 Поле ввода пароля

Для ввода пароля в HTML используется элемент `input` с атрибутом `type="password"`. Его отличительной чертой является то, что вводимые символы маскируются точками.

```
<h2> Введите пароль </h2>
```

```
<form><p> <input type="text" name="login" /></p> <p>
```

```
<input type="password" name="password" placeholder="пароль"/>
```

```
<input type="submit" value="Авторизация" /></form>
```

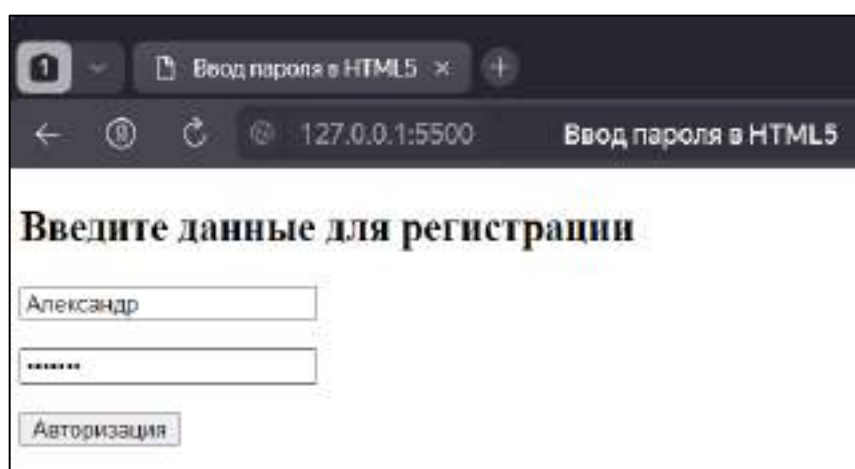


Рисунок 2.13 - Ввод пароля в HTML5

По своим атрибутам поле типа `password` не отличается от поля типа `text`. При использовании этого поля следует помнить, что пароль или любая другая информация, которая вводится в поле типа `password`, будет передаваться в виде ASCII-символа и будет доступна для просмотра.

2.6 Метки и автофокус

Вместе с полями ввода нередко используются метки, которые представлены элементом `label`. Метки создают аннотацию или заголовок к полю ввода, указывают,

для чего это поле предназначено. Для связи с полем ввода метка имеет атрибут `for`, который указывает на `id` поля ввода.

```
<body> <h1>Укажите цвет и форму дивана</h1> <form> <p>
    <label for="color">Цвет: </label>
    <input type="text" id="color" name="color" /> </p> <p>
    <label for="form">Форма: </label>
    <input type="text" id="form" name="form" /> </p> <p>
    <button type="submit">Отправить</button> </p>
</form> </body>
```

Так, текстовое поле здесь имеет атрибут `id="form"`. Поэтому у связанной с ним метки устанавливается атрибут `for="form"`. Нажатие на эту метку позволяет перевести фокус на текстовое поле для ввода фамилии (рисунок 2.14).

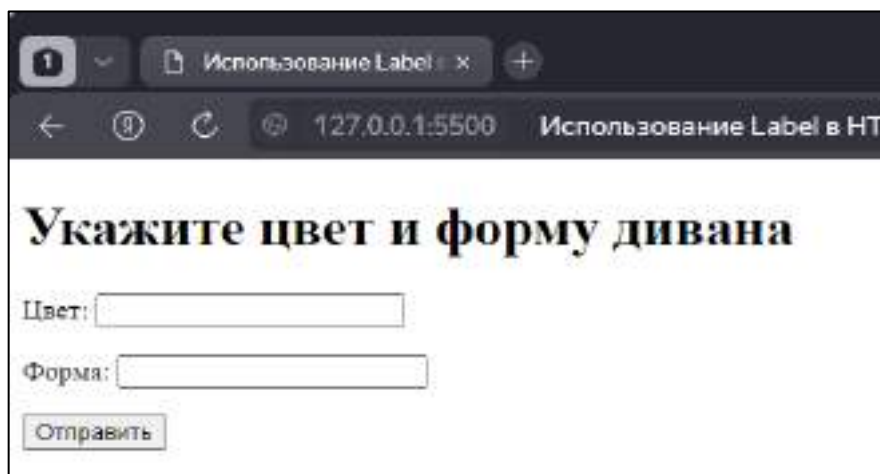


Рисунок 2.14 – Использование Label в HTML5

Собственно, на этом роль меток и заканчивается. Также можно установить автофокус по умолчанию на какое-либо поле ввода. Для этого применяется атрибут `autofocus`.

```
<h1>Укажите цвет и форму дивана</h1>
<form> <p> <label for="color">Цвет: </label>
    <input type="text" autofocus id="color" name="color" /> </p>
    <p> <label for="form">Форма: </label>
    <input type="text" id="form" name="form" /> </p>
    <p> <button type="submit">Отправить</button> </p>
</form>
```

Здесь при запуске страницы фокус сразу же переходит на текстовое поле (рисунок 2.15).

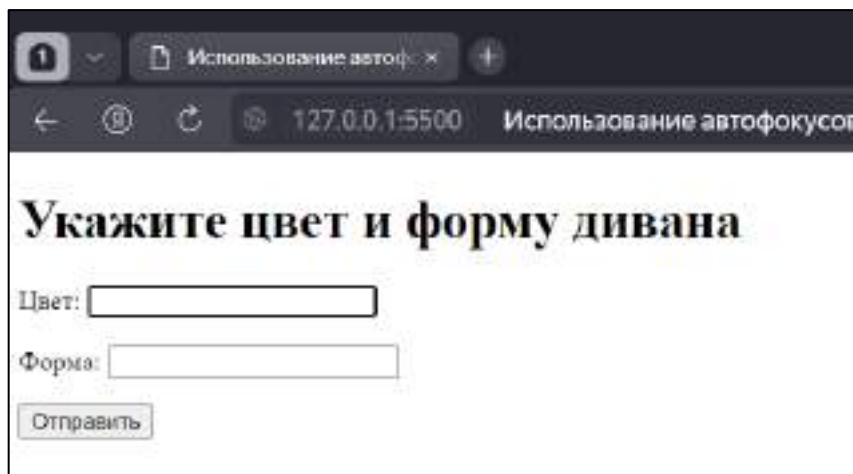


Рисунок 2.15 – Использование автофокусов в HTML

2.7 Элементы для ввода чисел

2.7.1 Простое числовое поле

Для ввода чисел в HTML5 используется элемент `input` с атрибутом `type="number"`. Он создает числовое поле, которое можно настроить с помощью следующих атрибутов:

- `min`: минимально допустимое значение;
- `max`: максимально допустимое значение;
- `readonly`: доступно только для чтения;
- `required`: указывает, что данное поле обязательно должно иметь значение;
- `step`: значение, на которое будет увеличиваться число в поле;
- `value`: значение по умолчанию.

Далее представлен пример работы с числовыми полями.

```
<body>
```

```
<h1>Укажите ваш возраст</h1>
```

```

<form><p> <label for="age">Возраст: </label>
  <input type="number" step="1" min="1" max="100" value="10" id="age"
name="age"/> </p>
  <p> <button type="submit">Отправить</button> </p> </form>
</body>

```

Реализация данного кода представлена на рисунке 2.16.

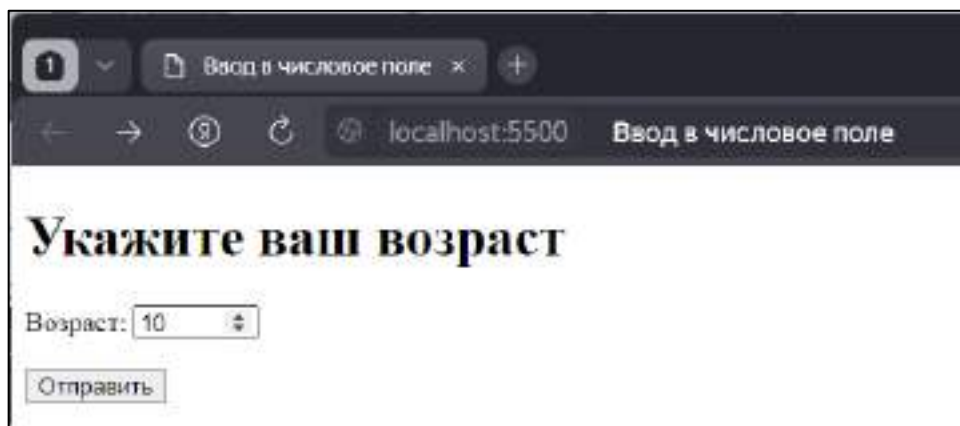


Рисунок 2.16 - Ввод в числовое поле

Здесь числовое поле по умолчанию имеет значение 10 (value="10"), минимально допустимое значение, которое можно ввести, - 1, а максимальное допустимое значение - 100. В зависимости от браузера визуализация этого поля может отличаться.

```

<body>
<h1>Сколько хотите купить мороженого?</h1>
<form>
<p> <label for="num">Количество: </label>
  <input type="number" list="numList"
step="10" min="30" max="100" value="100" id="num" name="num"/> </p>
  <p> <button type="submit">Готово</button> </p>
</form>
<datalist id="numList">
  <option value="1" /> <option value="2" />
  <option value="3" /> <option value="4" />
  <option value="5" /> </datalist> </body>

```

Пример реализации данного кода представлен на рисунке 2.17.

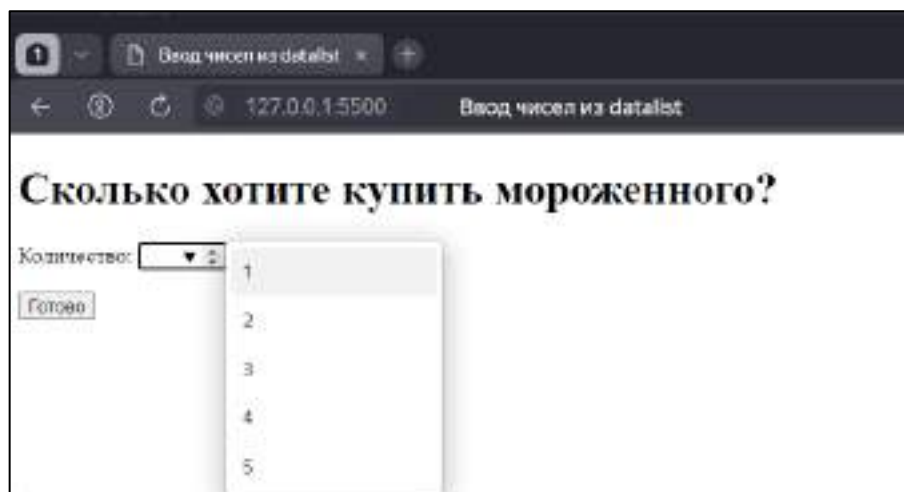


Рисунок 2.17 - Ввод чисел из datalist в HTML5

2.7.2 Ползунок

Ползунок представляет шкалу, на которой можно выбрать одно из значений. Для создания ползунка применяется элемент `input` с атрибутом `type="range"`. Во многом ползунок похож на простое поле для ввода чисел. Он также имеет атрибуты `min`, `max`, `step` и `value`.

```
<body> <form>
<p> <label for="num">Количество:</label>
1<input type="range" step="1" min="0" max="100" value="10" id="num"
name="num"/>100 </p>
<p> <button type="submit">Готово</button> </p> </form> </body>
```

Реализация данного кода представлен на рисунке 2.18.

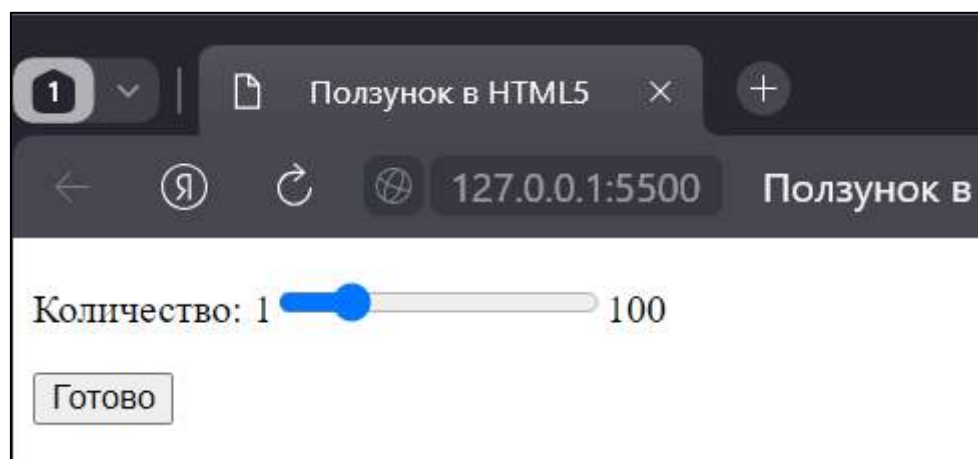


Рисунок 2.18 - Ползунок в HTML5 в Яндекс браузере

Аналогично, как и атрибут "number" тега input, атрибут "range" по-разному отображается в разных браузерах (рисунок 2.19).

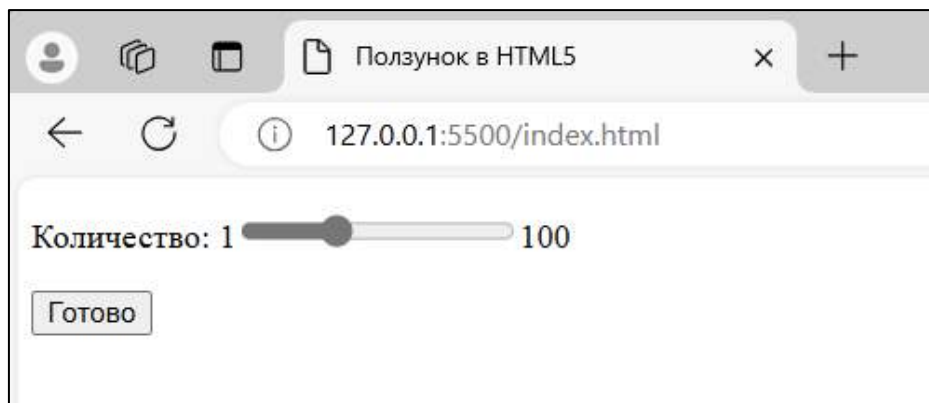


Рисунок 2.19 - Ползунок в HTML5 в Edge

2.8 Флажки и переключатели

2.8.1 Флажок

Флажок представляет элемент, который может находиться в двух состояниях: отмеченном и неотмеченном. Флажок создается с помощью элемента input с атрибутом type="checkbox".

```
<body>
<h2> Выберите любимого поэта</h2>
<form>
<p> <input type="checkbox" checked name="pushkin"/>Пушкин</p>
    <p> <input type="checkbox" value="lermontov" />
        <label for="lermontov">Лермонтов</label> </p>
    <p> <input type="checkbox" name="blok"/>Блок </p>
    <p> <button type="submit">Готово</button> </p>
</form>
</body>
```

Реализация данного кода представлена на рисунке 2.20.

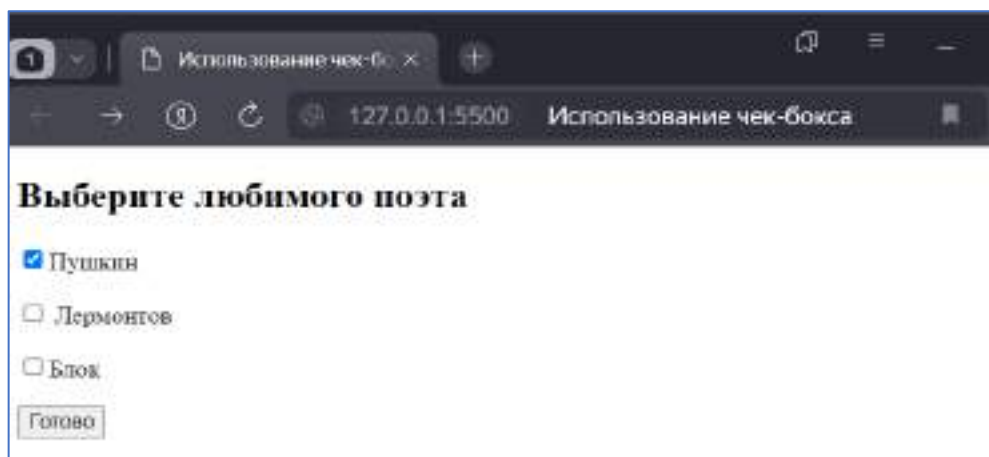


Рисунок 2.20 – Флажок (чек-бокс) в HTML5

Атрибут `checked` позволяет установить флажок в отмеченное состояние при загрузке страницы.

```
<input type="checkbox" value="lermontov" / checked>
```

2.8.2 Переключатели

Переключатели или радиокнопки похожи на флажки, они также могут находиться в отмеченном или неотмеченном состоянии. Только для переключателей можно создать одну группу, в которой одновременно можно выбрать только один переключатель, так как в отличие от переключателя используется одно имя для присвоения значения.

```
<body>
  <form> <h2>Укажите цвет</h2>
  <p> <input type="radio" value="white" checked name="color"/>белый </p> <p>
    <input type="radio" value="red" name="color"/>красный</p><h2>Выберите
цветы</h2><p>
  <input type="radio" value="rose" checked name="flower"/>Розы </p> <p>
  <input type="radio" value="tulip" name="flower"/>Тюльпаны</p> <p>
  <button type="submit">Готово</button>   </p>
</form>
</body>
```

Реализация данного кода представлена на рисунке 2.21.

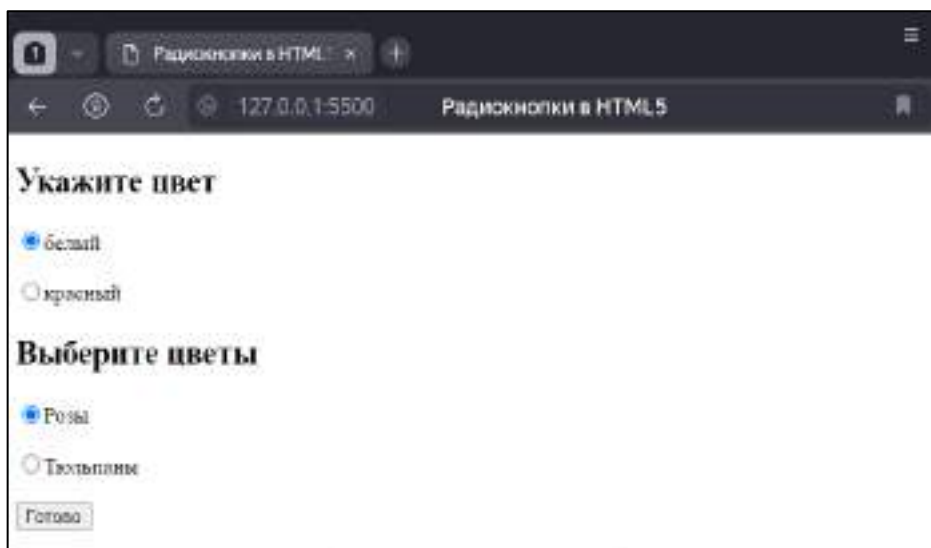


Рисунок 2.21 - Радиокнопки в HTML5

Для создания радиокнопки надо указать атрибут `type="radio"`. И теперь другой атрибут `name` указывает не на имя элемента, а на имя группы, к которой принадлежит элемент-радиокнопка. В данном случае две группы радиокнопок: `color` и `flower`. Из каждой группы можно выбрать только один переключатель. Опять же чтобы отметить радиокнопку, у нее устанавливается атрибут `checked`.

Важное значение играет атрибут `value`, который при отправке формы позволяет серверу определить, какой именно переключатель был отмечен.

2.9 Элементы для ввода цвета, url, email, телефона

2.9.1 Установка цвета

За установку цвета в HTML5 отвечает специальный элемент `input` с типом `color`.

```
<label for="tsvet">Выберите цвет</label>
```

```
<input type="color" id="tsvet" name="favcolor" />
```

Элемент отображает выбранный цвет. А при нажатии на него появляется специальное диалоговое окно для установки цвета (рисунок 2.22).

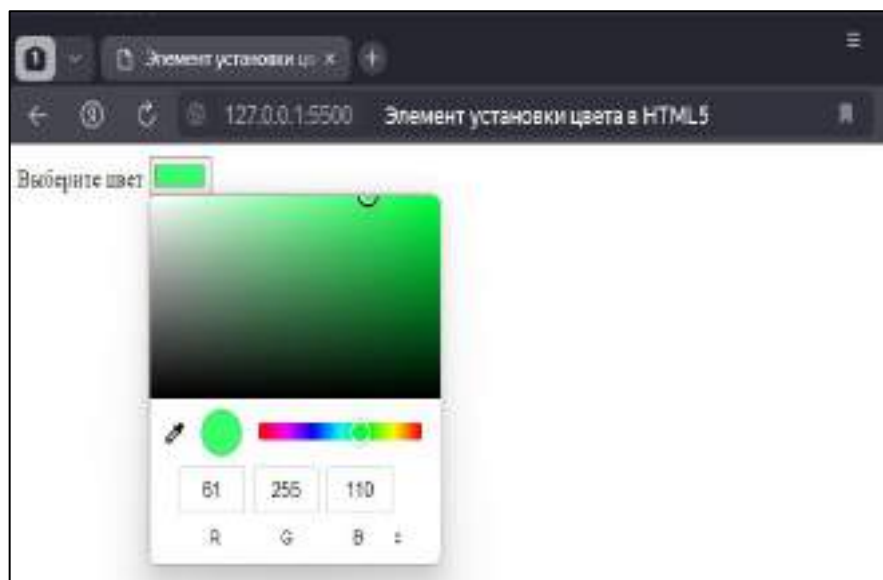


Рисунок 2.22 - Элемент установки цвета в HTML5

Значением этого элемента является числовой шестнадцатеричный код выбранного цвета. С помощью элемента `datalist` можно указать набор цветов, из которых пользователь сможет выбрать нужный.

```
<label for="tsvet">Выбор цвета</label>
<input type="color" list="colors" id="tsvet" name="tsvet" />
<datalist id="colors">
  <option value="#FF1493" label="DeepPink">
  <option value="#20B2AA" label="LightSeaGreen">
  <option value="#1E90FF" label="DodgerBlue">
</datalist>
```

Реализация данного кода представлена на рисунке 2.23.

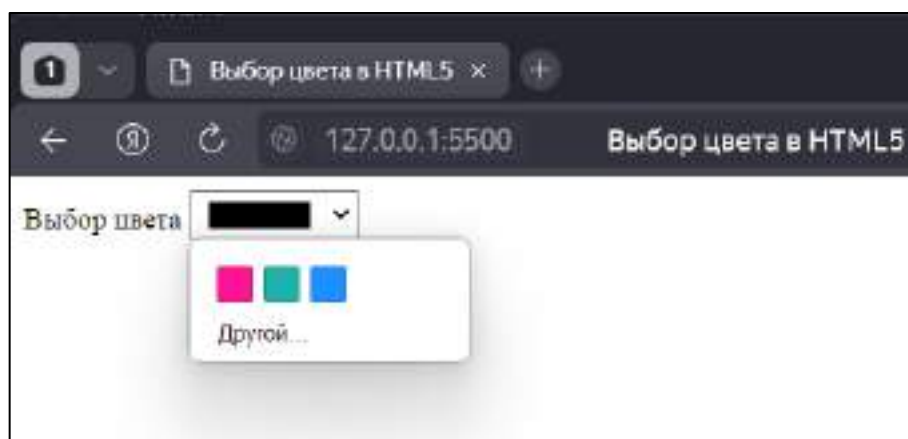


Рисунок 2.23 - Выбор цвета в HTML5

Каждый элемент option в datalist должен в качестве значения принимать шестнадцатеричный код цвета, например, "#0000FF". После выбора цвета данный числовой код устанавливается в качестве значения в элементе input.

2.9.2 Поля для ввода url, email, телефона

Различные поля ввода input предназначены для ввода таких данных, как URL-адрес, адрес электронной почты и номер телефона. Они однотипны и во многом отличаются только тем, что в качестве атрибута type принимают значения email, tel и url соответственно. Для их настройки можно использовать те же атрибуты, что и для обычного текстового поля:

- maxlength: максимальное количество символов, разрешенное в поле;
- pattern: определяет шаблон, которому должен соответствовать введенный текст;
- placeholder: определяет текст, который будет отображаться в поле по умолчанию;
- readonly: делает текстовое поле доступным только для чтения;
- required: указывает, что текстовое поле должно иметь значение;
- size: определяет ширину поля в видимых символах;
- value: определяет значение поля по умолчанию;
- list: определяет привязку к элементу datalist со списком возможных значений.

<body>

 <form>

 <p> <label for="email">Почта: </label><input type="email" placeholder="person@mail.ru" id="email" name="email"/> </p>

 <p> <label for="url">URL: </label><input type="url" id="url" name="url"/> </p>

 <p> <label for="phone">Телефон: </label>

 <input type="tel" placeholder="+7 (XXX) - XXX - XX - XX" id="phone" name="phone"/>

 </p> <p> <button type="submit">Готово</button>

 </p> </form> </body>

Реализация данного кода представлена на рисунке 2.24.

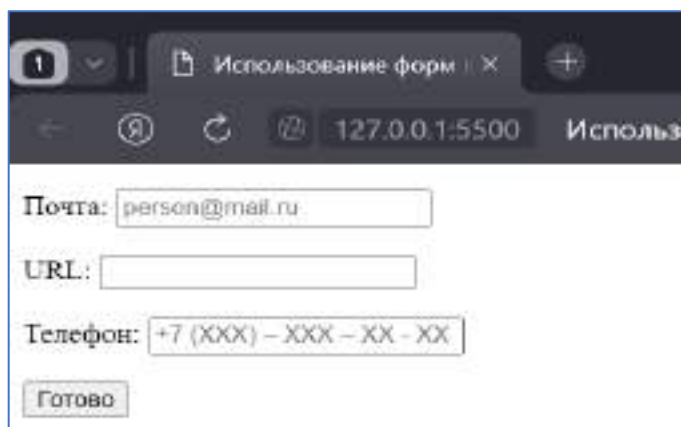


Рисунок 2.24 - Ввод почты, url, телефона в HTML5

Основное преимущество таких полей ввода перед обычными текстовыми полями заключается в том, что поля ввода электронной почты, URL-адреса и телефона используют соответствующий шаблон для проверки ввода. Например, если вводится неверное значение в какое-либо поле и отправляется форма, браузер может отобразить сообщение о неправильном вводе, а форма не будет отправлена (рисунок 2.25).

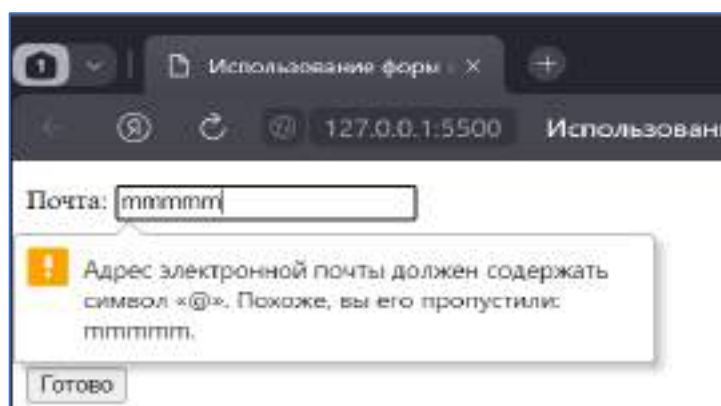


Рисунок 2.25 - Валидация почты в HTML5

2.10 Элементы для ввода даты и времени

Для работы с датами и временем в HTML5 предназначено несколько типов элементов `input`:

- datetime-local: устанавливает дату и время;
- date: устанавливает дату; month: устанавливает текущий месяц и год;
- time: устанавливает время; week: устанавливает текущую неделю.

Далее представлен пример установки даты.

```
<body><form> <p><label for="name"> ФИО: </label><input type="text" id="name"
name="name"/> </p>
<p><label for="date">Дата рождения: </label> <input type="date" id="date"
name="date"/></p> <p> <button type="submit">Готово</button> </p> </form> </body>
```

Реализация данного кода представлена на рисунке 2.26.

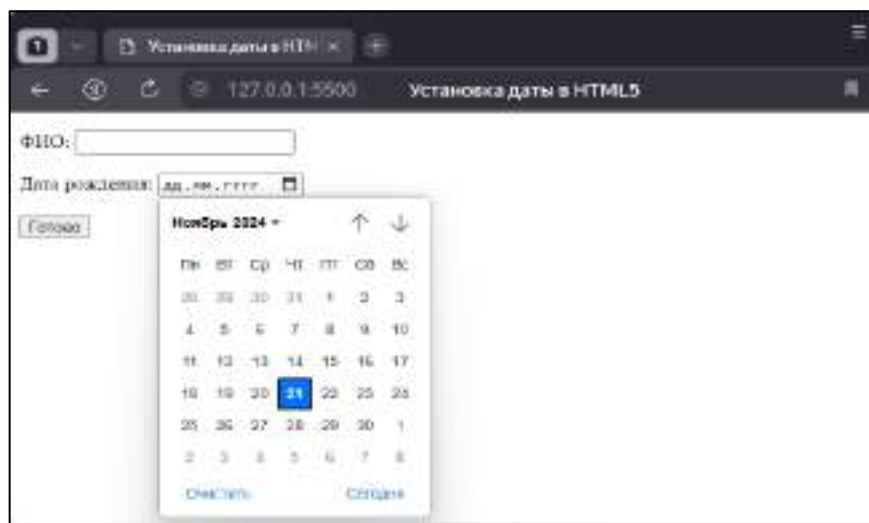


Рисунок 2.26 - Установка даты в HTML5

В данном случае при вводе в поле для даты будет открываться календарь. Аналогично многим расширениям HTML5 работа с датой по-разному отображается в разных браузерах.

Далее приведены примеры использования остальных элементов.

```
<body> <form>
<p> <label for="week">Неделя: </label><input type="week" name="week" id="week" /> </p>
<p> <label for="localdate">Дата и время: </label> <input type="datetime-local" id="localdate"
name="date"/></p><p>
<label for="month">Месяц: </label> <input type="month" id="month" name="month"/></p>
<p> <label for="time">Время: </label><input type="time" id="time" name="time"/></p>
<p><button type="submit">Готово</button></p> </form></body>
```

Реализация данного кода представлена на рисунке 2.27.

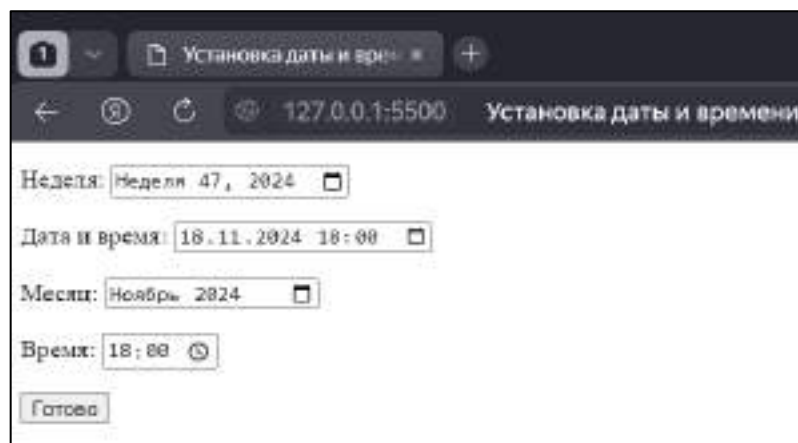


Рисунок 2.27 – Установка даты и времени в HTML5

При использовании этих элементов также надо учитывать, что Firefox поддерживает только элементы `date` и `time`, для остальных создаются обычные текстовые поля. А IE11 и вовсе не поддерживают эти элементы.

2.11 Отправка файлов

За выбор файлов на форме в HTML5 отвечает элемент `input` с атрибутом `type="file"`.

```
<body>
  <form      enctype="multipart/form-data"      method="post"
  action="http://localhost:5000/postfile.js">
    <p> <input type="file" name="file" /></p>
  <p><input type="submit" value="Готово" /> </p></form> </body>
```

Реализация данного кода представлена на рисунке 2.28.

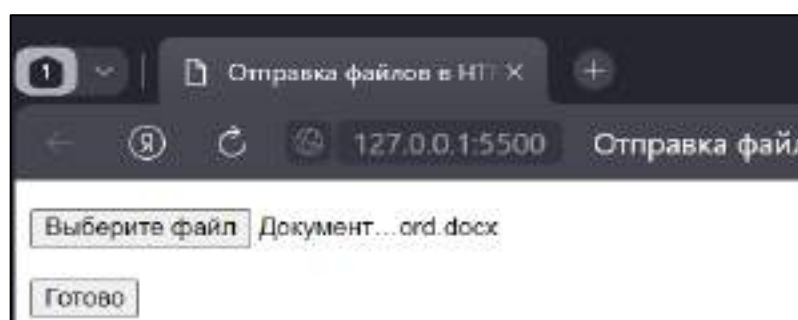


Рисунок 2.28 - Выбор файла в HTML5

При нажатии на кнопку “Выберите файл” открывается диалоговое окно для выбора файла. А после выбора рядом с кнопкой отображается имя выбранного файла. Для отправки файла на сервер форма должна иметь атрибут `enctype="multipart/form-data"`. С помощью атрибутов можно настроить элементы выбора файла:

- `accept`: устанавливает тип файл, которые допустимы для выбора;
- `multiple`: позволяет выбирать множество файлов;
- `required`: требует обязательной установки файла.

Далее приведен пример множественного выбора файлов.

```
<form                                enctype="multipart/form-data"                                method="post"
action="http://localhost:5000/postfile.js">
    <p> <input type="file" name="file" multiple /> </p>
    <p><input type="submit" value="Готово" /> </p> </form>
```

Реализация данного кода представлена на рисунке 2.29.

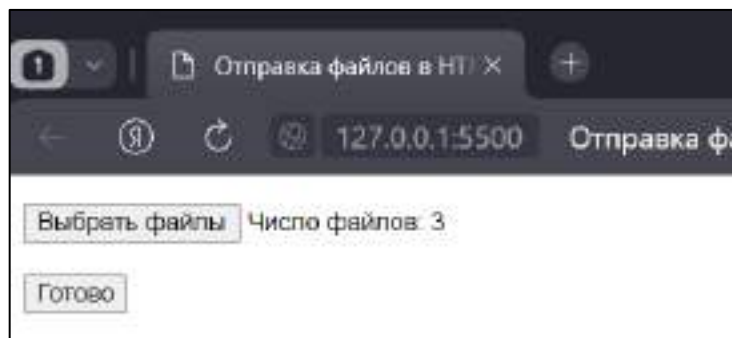


Рисунок 2.29 - Множественный выбор файлов в HTML5

При нажатии на кнопку открывается диалоговое окно для выбора файлов, зажав клавишу CTRL или Shift, можно выбрать несколько файлов, а после выбора рядом с кнопкой отобразится количество выбранных файлов:

2.12 Список select

Элемент `select` создает список. В зависимости от настроек это может быть выпадающий список для выбора одного элемента, либо раскрытый список, в котором

можно выбрать сразу несколько элементов. Далее представлен пример создания выпадающего списка.

```
<body> <form method="get"> <p>
  <label for="name">Наименование</label>
  <select id="name" name="name">
    <option value="pen">Ручка</option>
    <option value="pencil">Карандаш</option>
    <option value="eraser">Ластик</option>
    <option value="liner">Линейка</option> </select>
  </p> <p> <input type="submit" value="Готово" /> </p>
</form> </body>
```

Реализация данного кода представлена на рисунке 2.30.

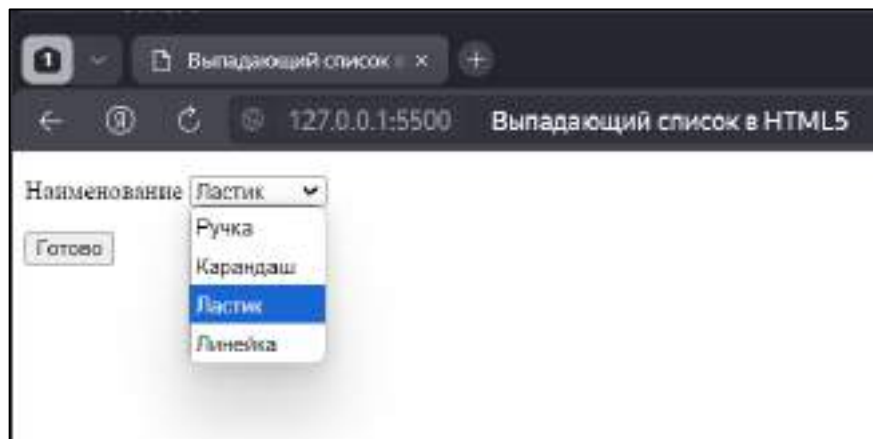


Рисунок 2.30 - Выпадающий список в HTML5

Внутри элемента `select` помещаются элементы `option` - элементы списка. Каждый элемент `option` содержит атрибут `value`, который хранит значение элемента. При этом значение элемента `option` не обязательно должно совпадать с отображаемым им текстом. Например:

```
<option value="pen">Ручка</option>
```

С помощью атрибута `selected` можно установить выбранный по умолчанию элемент - это необязательно должен быть первый элемент в списке:

```
<select id="name" name="name">
```

```
<option value="pen">Ручка</option>
```

```
<option value="pencil">Карандаш</option>
```

```

<option value="eraser">Ластик</option>
<option value="liner">Линейка</option> </select>
</select>

```

Для создания списка с множественным выбором к элементу select необходимо добавить атрибут multiple.

```

<body> <form method="get"> <p>
    <label for="name">Наименование техники:</label> <br/>
    <select multiple id="name" name="name">
        <option value="phone">Телефон</option>
        <option value="laptop">Ноутбук</option>
        <option value="plan">Планшет</option>
        <option value="laptop">Принтер</option>
        <option value="mouse">Мышь</option>
    </select> </p> <p> <input type="submit" value="Готово" /> </p> </form></body>

```

Зажав клавишу Ctrl, можно выбрать в таком списке несколько элементов:

Реализация данного кода представлена на рисунке 2.31.

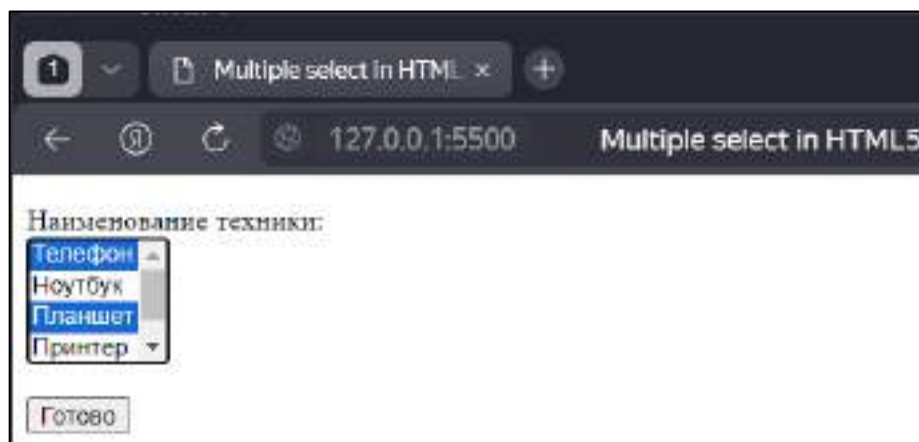


Рисунок 2.31 - Multiple select в HTML5

Select также позволяет группировать элементы с помощью тега <optgroup>. Далее представлен пример использования группировки.

```

<body> <form method="get"> <p>
    <label for="name">Наименование</label> <select id="name" name="name">
        <optgroup label="Канцтовары">
            <option value="pen">Ручка</option>
            <option value="pencil">Карандаш</option>

```

```

<option value="eraser">Ластик</option>
<option value="liner">Линейка</option> </select>
</optgroup><optgroup label="Учебники">
<option value="book">Учебник по математике</option> ">
<option value="book">Учебник по физике</option> </optgroup> </select> </p>
<p> <input type="submit" value="Готово" /></p> </form> </body>

```

Реализация данного кода представлена на рисунке 2.32.

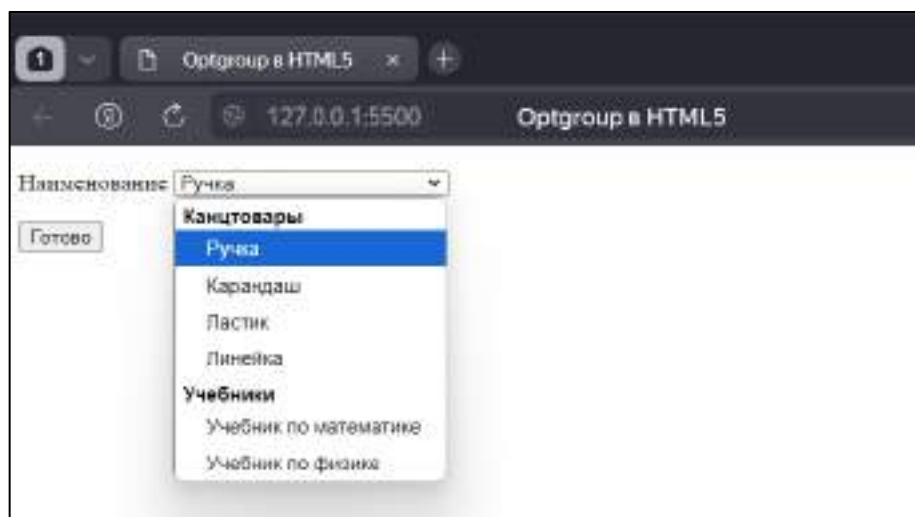


Рисунок 2.32 - Optgroup в HTML5

Использование групп элементов применимо как к выпадающему списку, так и к списку со множественным выбором.

2.13 Textarea

Элемент `<input type="text"/>` позволяет создавать простое однострочное текстовое поле. Однако возможностей этого элемента по вводу текста бывает недостаточно, и в этой ситуации можно использовать многострочное текстовое поле, представленное элементом `textarea`. С помощью дополнительных атрибутов `cols` и `rows` можно задать соответственно количество столбцов и строк.

Далее представлен пример использования тег `<textarea>`.


```

<body> <form method="get">
<p> <label for="comment">Контент</label><br/>
      <textarea id="comment" name="comment" placeholder="Текст поста"
      cols="30" rows="7"> </textarea></p>
<p> <input type="submit" value="Опубликовать" /></p>
</form> </body>

```

Реализация данного кода представлена на рисунке 2.33.

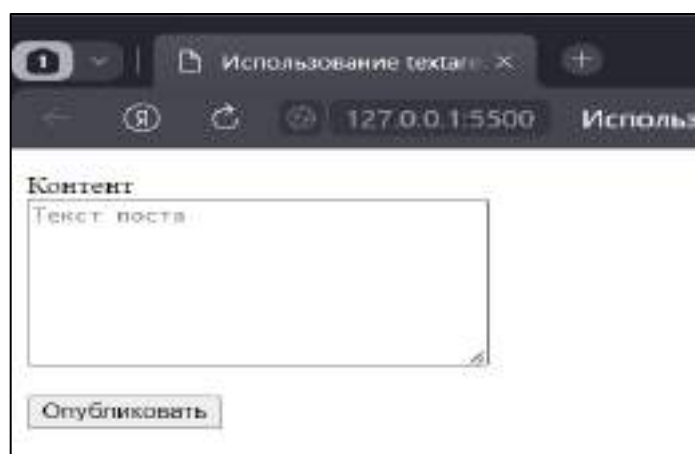


Рисунок 2.33 – Настройка размера блока textarea в HTML5

2.14 Валидация форм

Для создания форм в распоряжении имеются различные элементы, которые можно использовать. Можно вводить в них различные значения. Однако нередко пользователи вводят не совсем корректные значения: например, ожидается ввод чисел, а пользователь вводит буквы и т.д. И для предупреждения и проверки некорректного ввода в HTML5 существует механизм валидации.

Преимущество использования валидации в HTML5 заключается в том, что пользователь после некорректного ввода может сразу получить сообщение об ошибке и внести соответствующие изменения в введенные данные.

Для создания валидации у элементов форм HTML5 используются атрибуты:

–required: требует обязательного ввода значения. Для элементов textarea, select, input (с типом text, password, checkbox, radio, file, datetime-local, date, month, time, week, number, email, url, search, tel);

–min и max: минимально и максимально допустимые значения. Для элемента input с типом datetime-local, date, month, time, week, number, range;

–pattern: задает шаблон, которому должны соответствовать вводимые данные. Для элемента input с типом text, password, email, url, search, tel.

2.14.1 Атрибут required

Атрибут required требует обязательного наличия значения.

```
<body><form method="get"> <p> <label for="login">Почта:</label>  
    <input type="text" required id="login" name="login" />  
</p> <p> <label for="password">Пароль:</label>  
    <input type="password" required id="password" name="password" />  
</p> <p> <input type="submit" value="Готово" /> </p> </form> </body>
```

Реализация данного кода представлена на рисунке 2.34.

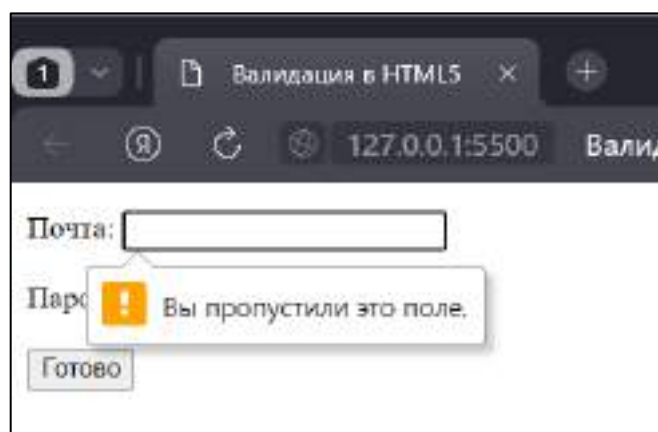


Рисунок 2.34 - Атрибут required в HTML5

Если не ввести в эти поля никаких данных, оставив их пустыми, и нажать на кнопку отправки, то браузер высветит сообщения об ошибке, а данные не будут отправлены на сервер. В зависимости от браузера визуализация сообщения может несколько отличаться. Также границы некорректного поля ввода могут окрашиваться

в красный цвет. Например, поведение при отправке пустых сообщений в Microsoft Edge.

2.14.2 Атрибуты max и min

Для ограничения диапазона вводимых значений применяются атрибуты max и min.

```
<body> <form method="get">  
  <p> <label for="age">Возраст:</label>  
  <input type="number" min="1" max="100" value="18" id="age" name="age"/> </p>  
  <p> <input type="submit" value="Готово" /> </p> </form> </body>
```

Реализация данного кода представлена на рисунке 2.35.

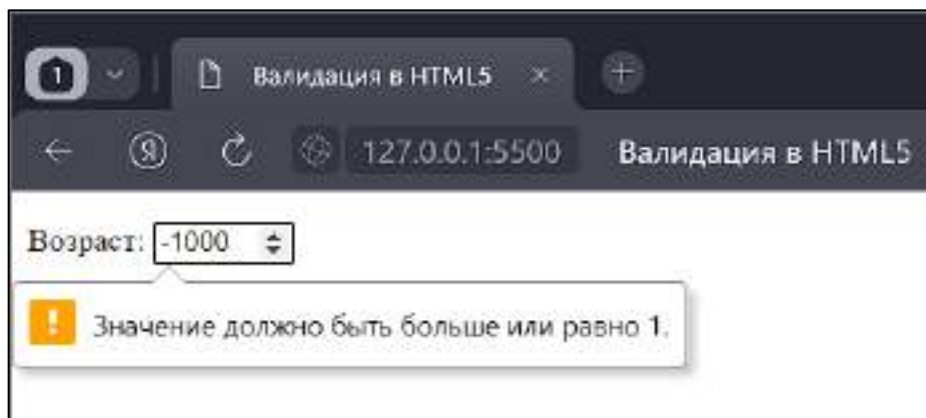


Рисунок 2.35 - Ограничение диапазона чисел в HTML5

2.14.3 Синтаксис регулярных выражений

2.14.3.1 Метасимволы

Регулярные выражения могут использовать метасимволы - символы, которые имеют определенный смысл:

- \d: соответствует любой цифре от 0 до 9;
- \D: соответствует любому символу, который не является цифрой;

- \w: соответствует любой букве, цифре или символу подчеркивания (диапазоны A-Z, a-z, 0-9);
- \W: соответствует любому символу, который не является буквой, цифрой или символом подчеркивания (то есть не находится в следующих диапазонах A-Z, a-z, 0-9);
- \s: соответствует пробелу;
- \S: соответствует любому символу, который не является пробелом;
- .: соответствует любому символу.

Здесь надо заметить, что метасимвол \w применяется только для букв латинского алфавита, кириллические символы для него не подходят.

Так, стандартный формат номера телефона +1-234-567-8901 соответствует регулярному выражению \d-\d\d\d-\d\d\d-\d\d\d\d. Например, числа номера заменяются нулями:

```
let phoneNumber = "+1-234-567-8901";
let myExp = /\d-\d\d\d-\d\d\d-\d\d\d\d/;
phoneNumber = phoneNumber.replace(myExp, "0000000000");
document.write(phoneNumber);
```

2.14.3.2 Модификаторы

Кроме выше рассмотренных элементов регулярных выражений есть еще одна группа комбинаций, которая указывает, как символы в строке будут повторяться. Такие комбинации еще называют модификаторами:

- {n}: соответствует n-ому количеству повторений предыдущего символа. Например, h{3} соответствует подстроке "hhh";
- {n,}: соответствует n и более количеству повторений предыдущего символа. Например, h{3,} соответствует подстрокам "hhh", "hhhh", "hhhhh" и т.д.;
- {n,m}: соответствует от n до m повторений предыдущего символа. Например, h {2, 4} соответствует подстрокам "hh", "hhh", "hhhh";

- `?`: соответствует одному вхождению предыдущего символа в подстроку или его отсутствию в подстроке. Например, `/h?ome/` соответствует подстрокам "home" и "ome";
- `+`: соответствует одному и более повторений предыдущего символа;
- `*`: соответствует любому количеству повторений или отсутствию предыдущего символа;

Например, берется номер тот же телефона. Ему соответствует регулярное выражение `\d-\d\d\d-\d\d\d-\d\d\d\d`. Однако с помощью выше рассмотренных комбинаций можно его упростить: `\d-\d{3}-\d{3}-\d{4}`

Также надо отметить, что так как символы `?`, `+`, `*` имеют особый смысл в регулярных выражениях, то чтобы их использовать в обычным для них значении (например, надо заменить знак плюс в строке на минус), то данные символы надо экранировать с помощью слеша:

```
let phoneNumber = "+1-234-567-8901";
let myExp = /\+\\d-\\d{3}-\\d{3}-\\d{4}/;
phoneNumber = phoneNumber.replace(myExp, "80000000000");
console.log(phoneNumber);
```

2.14.4 Атрибут pattern

Атрибут `pattern` задает шаблон, которому должны соответствовать данные. Для определения шаблона используется язык регулярных выражений.

```
<body> <form method="get">
  <p> <h2>Укажите телефон</h2>
  <label for="phone">Телефон:</label>
  <input type="text" placeholder="+7999-999-99-99"
    pattern="+\\d {4}-\\d {3}-\\d {2}-\\d {2}" id="phone" name="phone" /> </p>
  <p> <input type="submit" value="Готово" /> </p> </form> </body>
```

Здесь для ввода номера телефона используется регулярное выражение `+\\d {4}-\\d {3}-\\d {2}-\\d {2}`. Оно означает, что первым элементом в номере должен идти знак плюс `+`. Выражение `\\d` представляет любую цифру от 0 до 9. Выражение `\\d`

{3} означает три подряд идущих цифры, а \d {4} - четыре цифры подряд. То есть это выражение будет соответствовать номеру телефона в формате "+7999-000-00-00".

Если ввести данные, которые не соответствуют этому шаблону, и нажать на отправку, то браузер отобразит ошибку (рисунок 2.36).

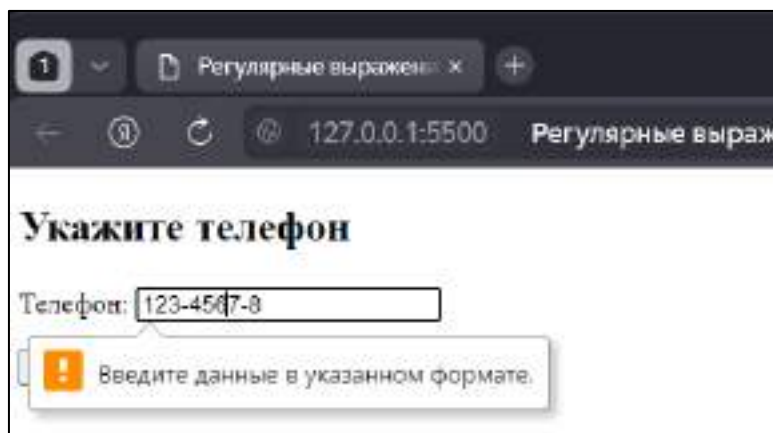


Рисунок 2.36 - Регулярные выражения в HTML5

2.14.5 Отключение валидации

Не всегда валидация является желаемой, иногда требуется ее отключить. И в этом случае можно использовать либо у элемента формы атрибут novalidate, либо у кнопки отправки атрибут formnovalidate.

```
<head> <body> <form novalidate method="get">
  <p> <label for="phone">Телефон:</label>
    <input type="text" placeholder="+1-234-567-8901"
      pattern="\+\\d-\\d{3}-\\d{3}-\\d{4}" id="phone" name="phone" /> </p>
  <p> <input type="submit" value="Готово" formnovalidate /> </p> </form> </body>
```

2.14.6 Элементы fieldset и legend

Для группировки элементов формы нередко применяется элемент fieldset. Он создает границу вокруг вложенных элементов, как бы создавая из них группу. Вместе с ним используется элемент legend, который устанавливает заголовок для группы элементов [14].

```

<body> <h2>Вход на сайт</h2>
<form> <fieldset> <legend>Введите данные:</legend>
<label for="login">Логин:</label><br>
<input type="text" name="login" id="login" /><br>
<label for="password">Пароль:</label><br>
<input type="password" name="password" id="password" /><br>
<input type="submit" value="Авторизация"> </fieldset></form> </body>

```

Реализация данного кода представлена на рисунке 2.37.

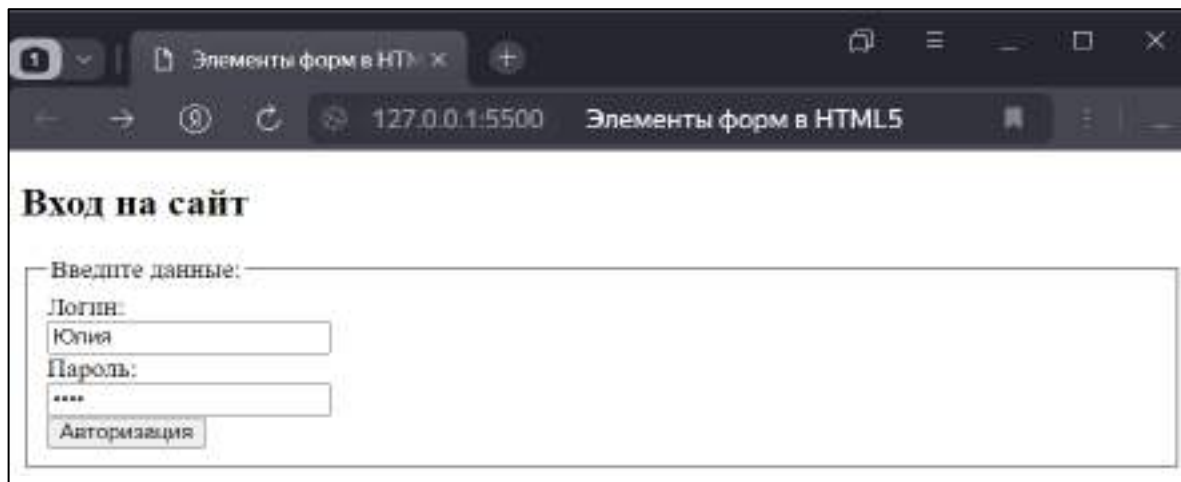


Рисунок 2.37 - Группировка элементов формы в HTML5

При необходимости можно создать на одной форме несколько групп с помощью элементов `fieldset`.

При создании форм на сайтах используется сочетание рассмотренных элементов управления формами. Далее представлен пример использования всех элементов при создании анкеты студента.

```

<form> <legend>Форма</legend>
<label for="login">*Фамилия:</label><br />
<input type="text" name="name" id="name" required /><br />
<label for="login">*Имя:</label><br />
<input type="text" name="surname" id="surname" required /><br />
<label for="login">Отчество:</label><br />
<input type="text" name="patronymic" id="patronymic" /><br /> <hr/>
<label for="sex">*Пол:</label><br />
<input type="radio" value="man" checked name="pol" />мужской

```

```

<input type="radio" value="woman" name="pol" />женский <hr/>
<p> <label for="date">*Дата рождения: </label>
    <input type="date" id="date" name="date" required /> </p><hr />
<p> <label for="email">*Почта: </label>
    <input type="email" placeholder="person@mail.ru" id="email" name="email"
        required /> </p><hr />
<label for="phone">*Телефон:</label>
<input type="text" pattern="\+7[0-9]{10}" placeholder="+7_____"
    id="phone" name="phone" required/><br /><hr />
<p><label for="name">*Ваш курс:</label>
    <select id="name" name="name" required>
        <option value="1 курс">Первый</option>
        <option value="2 курс">Второй</option>
        <option value="3 курс">Третий</option>
        <option value="4 курс">Четвертый</option> </select> </p><hr />
<label>Выберете направления подготовки:</label><br />
<input type="checkbox" name="blok" />ПиНЖ<br />
<input type="checkbox" name="blok" />ИВТ<br />
<input type="checkbox" name="blok" />САПР<br />
<p> <label for="mastery">Уровень освоения программирования:</label>
    1<input type="range" step="1" min="0" max="10" value="5" id="mastery"
name="mastery"/>10 </p>
<p> <label for="grade">Средняя оценка по предметам</label>
    <input type="number" step="0.1" min="1" max="5" value="5" id="grade"
name="grade"/> </p>
<label for="comment">О себе:</label><br />
<textarea id="comment" name="comment" placeholder="Расскажите о себе..."
    cols="30" rows="3" ></textarea> <hr />
<label for="login">*Логин:</label><br />
<input type="text" name="login" id="login" required /><br /><hr />
<label for="password">*Пароль:</label><br />
<input type="password" name="password" id="password" required /><br /> <hr />
<input type="submit" value="Отправить" /> </fieldset> </form>

```

Реализация данного кода представлена на рисунке 2.38.

Использование форм : x

localhost:5500 Использование форм в Н...

Форма

*Фамилия: Белов

*Имя: Евгений

Отчество: Александрович

*Пол: ☒ мужской ☐ женский

*Дата рождения: 01.05.2002

*Почта: evgen2@mail.ru

*Телефон: +79991234567

*Ваш курс: Четвертый

Выберете направления подготовки:

☒ ПИИЖ

☐ ИВТ

☐ САПР

Уровень освоения программирования: 1 10

Средняя оценка по предметам 4.2

О себе: Студант 4го курса. Хорошист

*Логин: Evgen12

*Пароль: *****

Отправить

Рисунок 2.38 – Пример использования всех элементов управления формами

2.15 Контрольные вопросы

- 1) Отправка данных с помощью тега `form`, основные атрибуты.

- 2) Описать работу с однострочными и многострочными текстовыми полями, списками, флажками и радиокнопками.
- 3) Задание элементов форм для ввода цвета, url, email, телефона.
- 4) Рассказать об элементах форм для ввода даты и времени.
- 5) Валидация форм. Перечислить основные инструменты валидации.

2.16 Тестирование по главе

- 1) Атрибут ... задает тип кнопки для button
 - а) submit;
 - б) button;
 - в) type;
 - г) form.
- 2) Атрибут ... устанавливает текст по умолчанию
 - а) dir;
 - б) pattern;
 - в) placeholder;
 - г) readonly.
- 3) Метки представлены элементом ...
 - а) label;
 - б) input;
 - в) meta;
 - г) form.
- 4) Атрибут ... указывает, что поле должно иметь значение
 - а) step;

- б) value;
- в) min;
- г) required.

5) Флажок создается при помощи элемента ...

- а) input;
- б) button;
- в) form;
- г) label.

6) Преимуществом каких полей является проверка введенных данных на корректность?

- а) текстовые поля;
- б) email;
- в) элемент ввода времени;
- г) формы.

7) Какой браузер поддерживает только элементы date и time?

- а) Yandex;
- б) Opera;
- в) Microsoft Edge;
- г) Firefox.

8) Элемент ... создает список

- а) select;
- б) option;
- в) input;
- г) label.

9) В чем отличие textarea от <input type="text"/>?

- б) отличий нет;
- в) `textarea` создает многострочное поле;
- г) `<input type="text"/>` создает многострочное поле;
- д) в `textarea` можно вставлять картинки.

10) Какой символ в синтаксисе регулярных выражений соответствует концу строки?

- а) “?”;
- б) “*”;
- в) “\$”;
- г) “.”.

3 Семантическая верстка страниц в HTML5

Семантическая верстка - подход к разметке, который опирается не на содержание сайта, а на смысловое предназначение каждого блока и логическую структуру документа. Семантическая разметка страниц помогает поисковым ботам лучше понимать, что находится на странице, и в зависимости от этого ранжировать сайты в поисковой выдаче.

Среди “старых” тегов из ранних версий HTML тоже есть семантические - например, тег `<p>`, который обозначает параграф. Но в актуальной версии HTML 5 есть семантические теги почти для всех основных частей сайта, и лучше пользоваться ими. Вот несколько примеров семантических тегов.

1) `<article>`. Значение: независимая, отделяемая смысловая единица, например, комментарий, твит, статья, виджет ВК.

2) `<section>`. Значение: смысловой раздел документа. Неотделяемый, в отличие от `<article>`.

3) `<aside>`. Значение: побочный, косвенный для страницы контент. Может иметь свой заголовок. Может встречаться несколько раз на странице.

4) `<nav>`. Значение: навигационный раздел со ссылками на другие страницы или другие части страниц. Используется для основной навигации, а не для всех групп ссылок.

5) `<header>`. Значение: вводная часть смыслового раздела или всего сайта, обычно содержит подсказки и навигацию. Чаще всего повторяется на всех страницах сайта. Этих элементов может быть несколько на странице.

6) `<main>`. Значение: основное, не повторяющееся на других страницах, содержание страницы. Особенности: должен быть один на странице, исходя из определения.

7) `<footer>`. Значение: заключительная часть смыслового раздела или всего сайта, обычно содержит информацию об авторах, список литературы, копирайт и так далее.

3.1 Элемент article

Элемент `article` представляет собой полный блок информации на странице, который можно просматривать отдельно и использовать независимо от других блоков. Например, это может быть сообщение на форуме, статья в блоге, интернет-журнале или комментарий пользователя.

Элемент `article` может содержать несколько элементов `article`. Например, вы можете обернуть всю запись блога в элемент `article`, и этот элемент будет содержать другие элементы записи, представляющие комментарии к этой записи блога. То есть сообщение в блоге может рассматриваться как отдельная смысловая единица, и в то же время комментарии также могут рассматриваться отдельно, независимо от остального контента. Использование `article` на примере статьи из блога с комментариями.

```
<body>
<article>
  <h2> Интересные факты о дельфинах</h2><div>
    <article style="background-color: SteelBlue;">
      <p>Дельфины - удивительные морские млекопитающие, которые издавна вызывают
интерес и восхищение человека. Эти быстрые, сообразительные и дружелюбные создания по-
прежнему хранят много тайн и загадок для ученых. Мы собрали для вас подборку самых
любопытных фактов о жизни дельфинов, которые помогут лучше узнать их: </p>
      <ol>
        <li> Дельфины спят, поочередно отключая половины мозга.
        <li>Самки дельфинов кормят детенышей очень питательным молоком, которое содержит
до 10% жира.
        <li>У дельфинов нет никакого обоняния, зато они компенсируют это острым зрением.</li>
      </ol>
    </article>
    <div><h3>Комментарии</h3></div>
    <h4>Пользователь 1</h4> <p>Интересная статья про дельфинов. Открыл заново для
себя этих прекрасных животных. Обязательно поеду на экскурсию.
    </p> </div>
```

```
<div> <h4>Пользователь 2</h4>
```

```
<p>Хорошая статья. Много нового узнал про дельфинов. Автор все интересно  
рассказал.</p></div> </body>
```

Реализация данного кода представлена на рисунке 3.1.

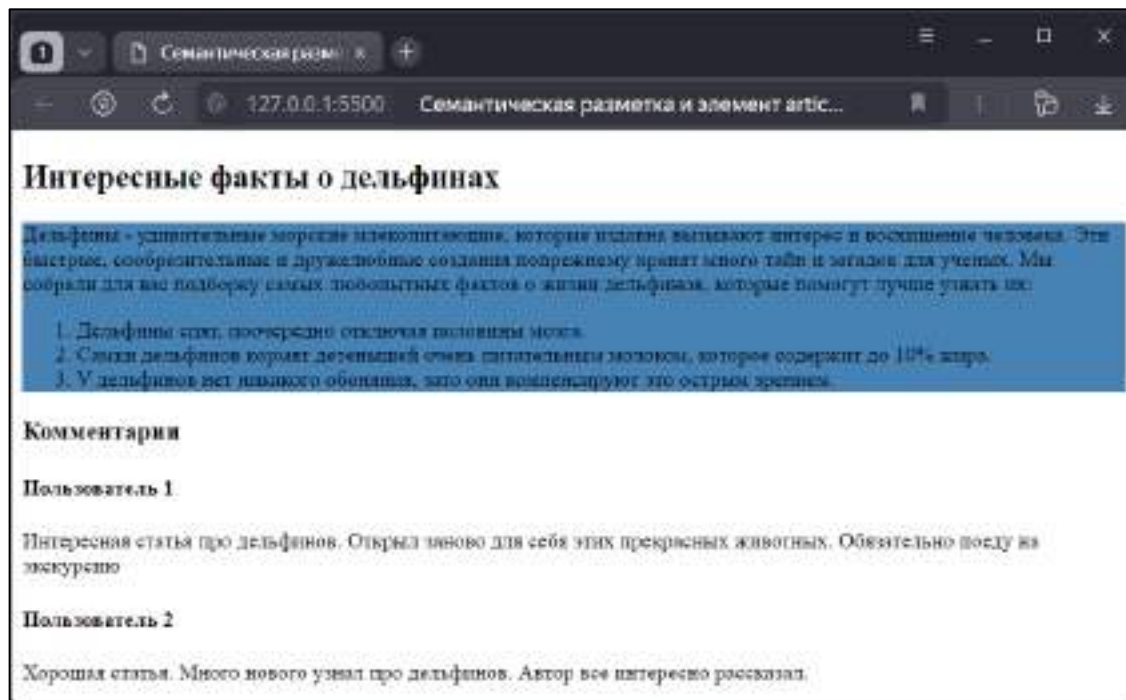


Рисунок 3.1 - Семантическая разметка и элемент article в HTML5

Здесь вся статья может быть помещена в элемент `article`, и в то же время каждый отдельный комментарий также представляет отдельный элемент `article`.

При использовании `article` надо учитывать, что каждый элемент `article` должен быть идентифицирован с помощью включения в него заголовка `h1-h6`.

3.2 Элементы `section`

Элемент раздела `section` объединяет и группирует связанную информацию в документе HTML. Например, раздел может содержать ряд вкладок на странице, новости, сгруппированные по категориям и т. д. Каждый элемент раздела должен быть идентифицирован заголовком `h1-h6`.

В этом случае элемент раздела может содержать несколько элементов article и группировать их, один элемент article может содержать несколько элементов раздела.

```
<body>
<article> <section style=" background-color:MediumSeaGreen;">
    <h2>Курорты Турции</h2>
</section>
<section style=" background-color:Turquoise;">
    <h3>Прекрасная Турция представляет множество курортов для туристов
со всего мира. К ним относятся красивейшие места на побережье Средиземного
моря.</h3> </section> </article>
<ul style="background-color:Aquamarine">
    <li>Кемер – морской курорт на юге средиземноморского побережья Турции. В
этой части Турецкой Ривьеры расположены галечные пляжи и большая пристань для яхт.
    <li>Белек – город в Турции на южном побережье Средиземного моря. Он
известен своими пляжами, термальными источниками и полями для гольфа. В
окрестностях Белека можно увидеть руины римского амфитеатра и колоннады
древнегреческого города Перге.
    <li>Аланья – курортный город в центральной части средиземноморского
побережья Турции, которое часто называют Турецкой Ривьерой. Город славится
своими окруженными отелями широкими пляжами, среди которых пляж Клеопатры, где
по легенде купалась египетская царица. </ul> </body>
```

Реализация данного кода представлена на рисунке 3.2.

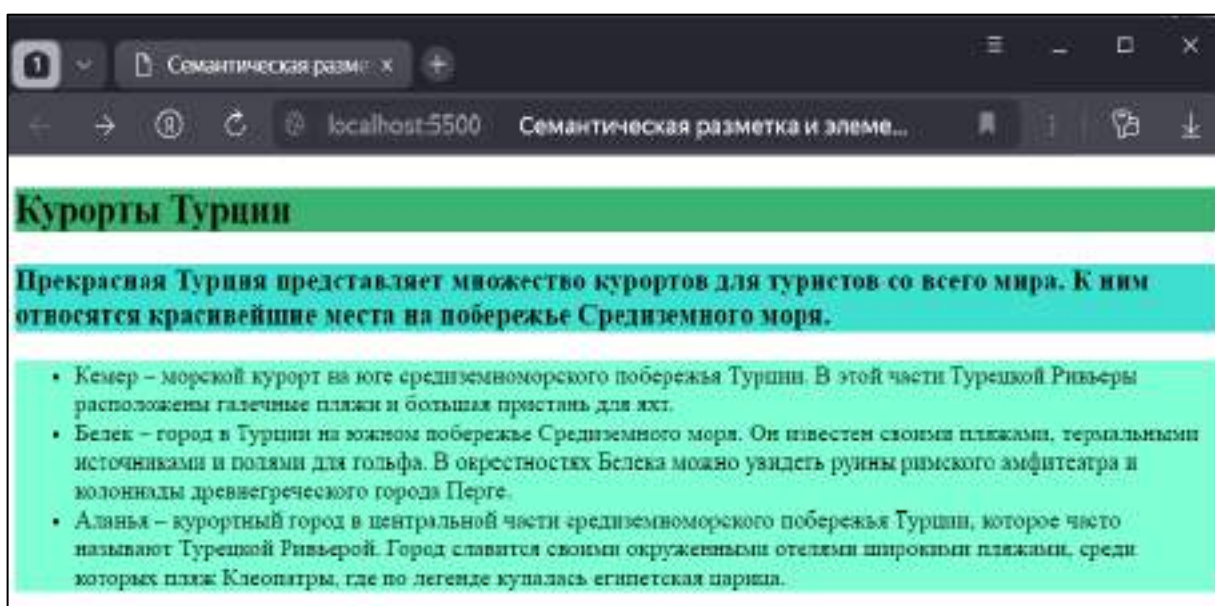


Рисунок 3.2 - Семантическая разметка и элемент Section в HTML5

3.3 Элемент nav

Элемент nav предназначен для размещения элементов навигации по сайту. Обычно это нумерованный список с набором ссылок. Можно использовать несколько элементов навигации на одном веб-сайте. Например, один элемент навигации используется для навигации по страницам веб-сайта, а другой - для навигации по HTML-документу.

Не все ссылки необходимо размещать в элементе nav. Некоторые ссылки могут не представлять связанный блок навигации, например, ссылка на домашнюю страницу, ссылка на лицензионное соглашение о предоставлении услуг и подобные ссылки, которые часто находятся внизу страницы. Обычно их достаточно определить в элементе footer, и нет необходимости предоставлять для них элемент навигации. Далее представлен пример использования элемента nav для создания навигационного меню.

```
<body>
  <nav style="background-color: #b955d2;">
    <header> <h2> Золотой век русской поэзии </h2>
      <p>Настоящими шедеврами русской поэзии являются произведения поэтов 18-19
веков. </p> </header>
      <ul> <li><a href="#part1"> Пушкин А.С.</a></li>
        <li><a href="#part2">Лермонтов М.Ю.</a></li>
        <li><a href="#part3">Тютчев Ф.И. </a></li> </ul> </nav>
    <section style="background-color:#d96fd5;" id="part1">
      <h4>Пушкин А.С.</h4> <p>Александр Пушкин начал писать свои первые произведения
уже в семь лет. Он создавал не только лирические стихи, но и сказки, историческую прозу и
произведения в поддержку революционеров - за вольнодумство поэта даже отправляли в
ссылки.</p>
    </section>
    <section style="background-color:#d96fd5;" id="part2">
      <h4>Лермонтов М.Ю.</h4> <p>Михаил Лермонтов - один из самых известных русских
поэтов, и признание к нему пришло еще при жизни. Его творчество, в котором сочетались
```

острые социальные темы с философскими мотивами и личными переживаниями, оказало огромное влияние на поэтов и писателей XIX–XX веков.

Тютчев Ф.И. Много лет Федор Тютчев был дипломатом и работал за границей, а стихи писал в свободное от службы время. Его произведения почти не печатали в России. Слава пришла к поэту после публикаций в журнале “Современник”.

Культура России.рф | **Контакты: E-mail cultrRF@mail.ru** © Copyright 2024

Реализация данного кода представлена на рисунке 3.3.

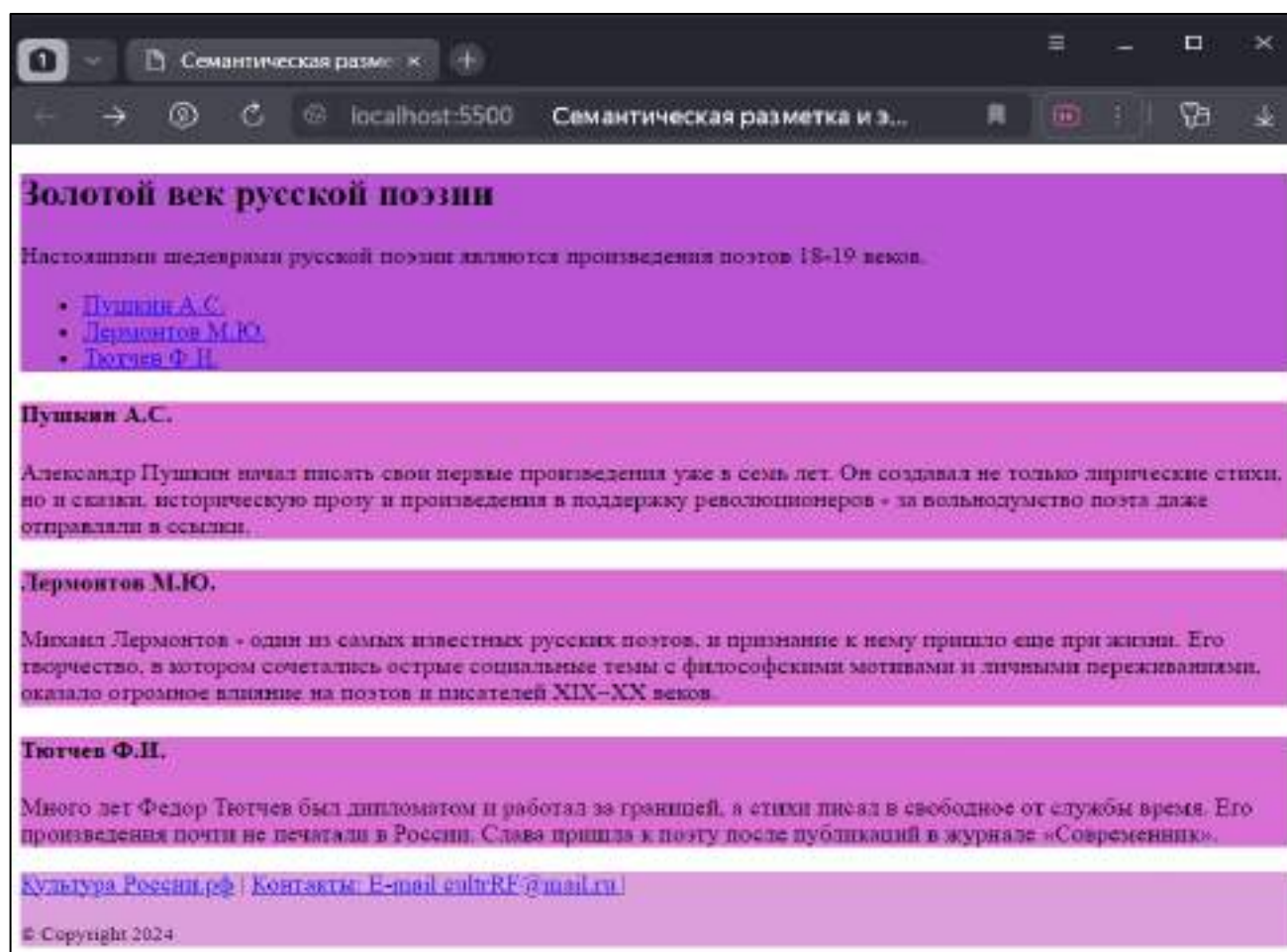


Рисунок 3.3 - Элемент nav в HTML5

В данном случае определены два блока nav - один для межстраничной навигации, а другой - для навигации внутри страницы.

3.4 Элемент header

Элемент header является как бы вводным элементом, предваряющим основное содержимое. Здесь могут быть заголовки, элементы навигации или какие-либо другие вспомогательные элементы, например, логотип, форма поиска и т.п. [15].

```
<body> <header style="background-color: #D8BFD8; padding-bottom: 1px;">
<h1>Онлайн-магазин цветов</h1> <nav> <ul>
<li><a href="/rose">Розы</a> <li><a href="/tulips">Тюльпаны</a> </ul></nav> </header>
<div>Информация о новинках магазина.... </div></body></html>
```

Реализация данного кода представлена на рисунке 3.4.

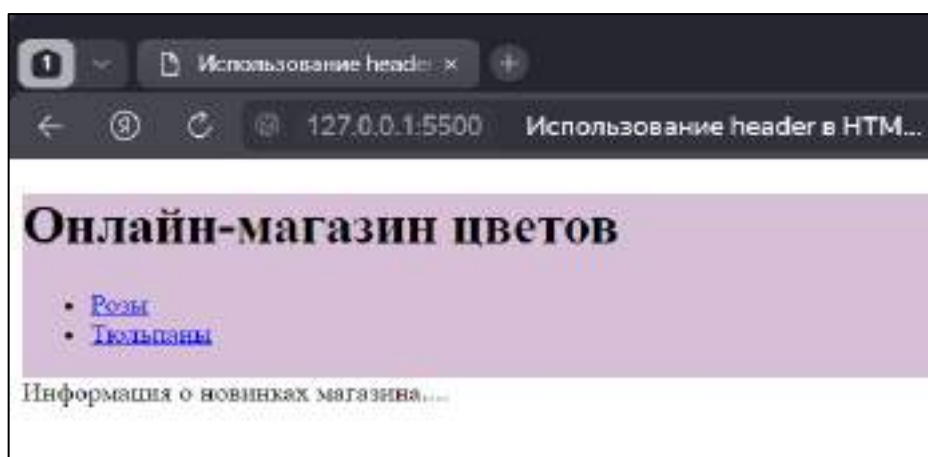


Рисунок 3.4 – Использование header в HTML5

3.5 Элемент footer

Элемент footer обычно содержит информацию о том, кто автор контента на веб-странице, копирайт, дата публикации, блок ссылок на похожие ресурсы и т.д. Как правило, подобная информация располагается в конце веб-страницы или основного содержимого, однако, footer не имеет четкой привязки к позиции и может использоваться в различных местах веб-страницы.

`<body> <section style="background-color: gold;"><h2>Магазин косметики для кошек Мяутуаль</h2>`

`<p>Профессиональная косметика для животных помогает убрать неприятный запах, оздоровить шерсть, сделать ее гладкой и шелковистой. В отличие от обычных средств для людей, она не вызывает аллергии, так как разработана специально с учетом особенностей кошек.</p>`

`<h3>Профессиональная косметика включает в себя:</h3>`

`Шампуни для кошек.`

`Парфюмерия.`

`Пудры и спреи для придания шерсти объёма.`

`Аксессуары для расчесывания.</section>`

`<footer style="background-color: khaki;">`

`<p> Наш адрес: г. Москва, ул. Мурчания, д.15</p>`

`<p> Наш почта: catmur@mail.ru</p>`

`Copyright © 2024. Murlika_cat.com</p> </footer> </body>`

Реализация данного кода представлена на рисунке 3.5.

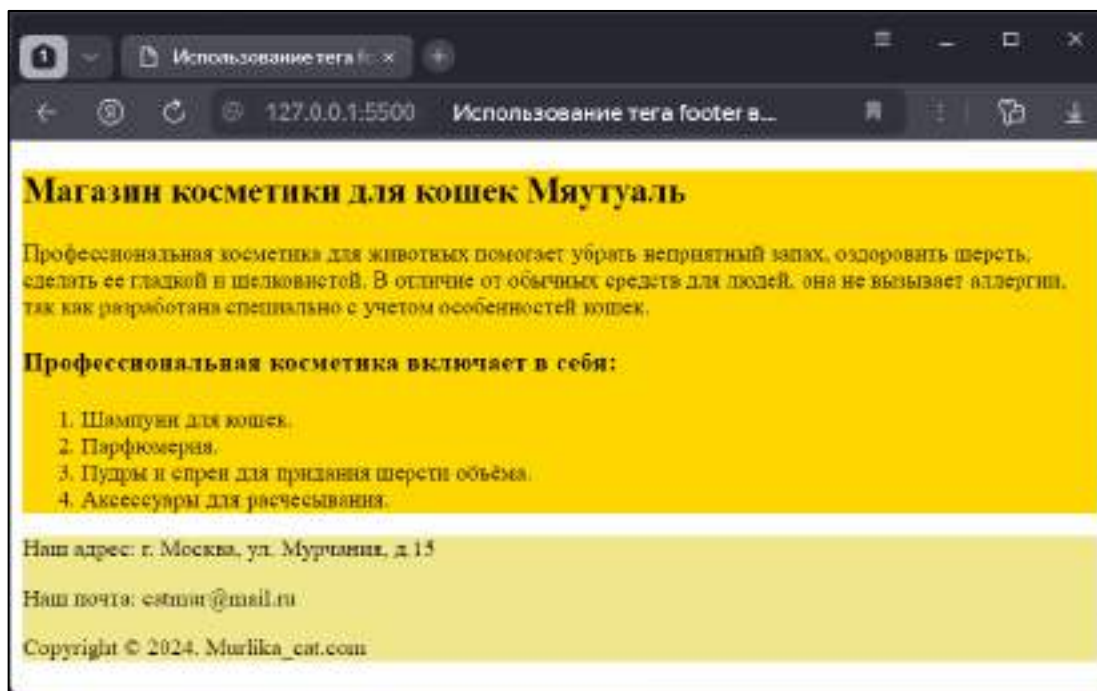


Рисунок 3.5 – Использование тега footer в HTML5

Здесь определен футер, в котором помещена информация об адресе, почте, информация о копирайте. Футер необязательно должен быть определен для всей страницы. Это может быть и отдельная секция контента.

3.6 Элемент address

Элемент address предназначен для отображения контактной информации, которая связана с ближайшим элементом article или body. Нередко данный элемент размещается в футере.

```
<body> <section style="background-color: MediumPurple;">
```

```
<h1>Stylist56</h1>
```

```
<p>Профессиональные стилисты-парикмахеры нашего салона красоты приглашают  
всех желающих выглядеть стильно, модно и неповторимо. Мы поможем сделать ваш имидж  
оригинальным и абсолютно эксклюзивным. Каждому клиенту мастера подберут подходящую  
прическу с учетом современных тенденций и индивидуальных предпочтений. Специалисты  
работают с волосами любой длины и структуры. Выполняются модельные стрижки для  
мужчин и женщин согласно типу внешности, возрасту и комплекции. Осуществляем  
завивку и укладку волос различной длины.</p>
```

```
</section>
```

```
<footer style="background-color: Violet;">
```

```
<address><p><small> Наш адрес: г. Оренбург, ул. Новая, д.7</p>
```

```
<p> Электронная почта: stylist56@mail.ru</p>
```

```
Copyright &copy; 2024. Stylist56Orenburg</small></p></address>
```

```
</footer> </body>
```

Реализация данного кода представлена на рисунке 3.6.

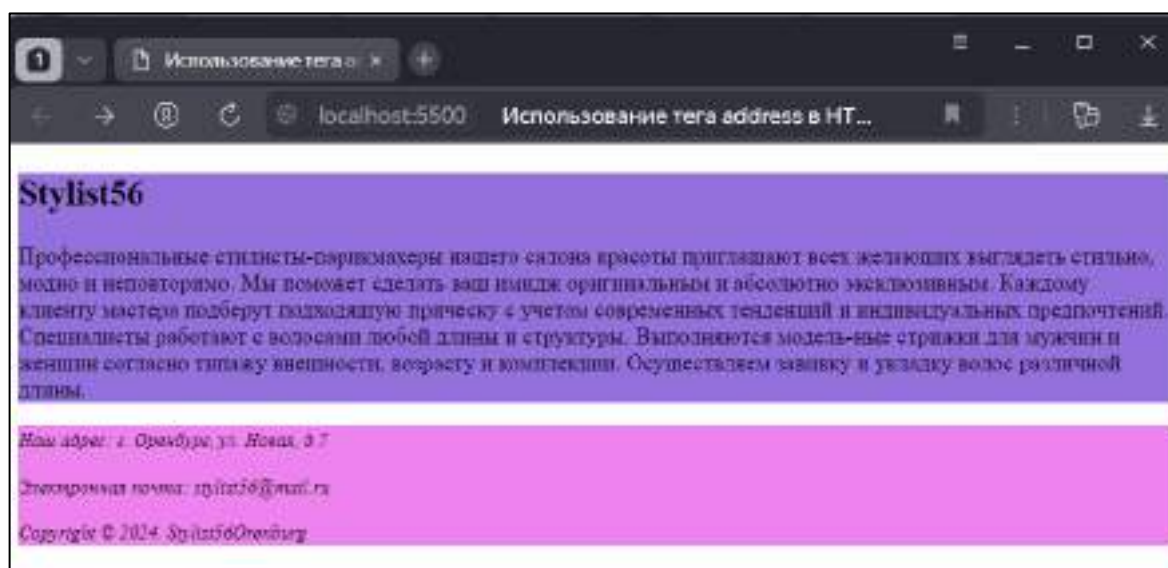


Рисунок 3.6 – Использование тега address в HTML5

3.7 Элемент aside

Элемент `aside` представляет собой контент, который косвенно связан с остальным контентом веб-сайта и может отображаться независимо от него. Этот элемент можно использовать, например, для боковых панелей, для рекламных блоков, блоков элементов навигации, различных плагинов.

```
<body>
<div style="display: flex; justify-content: space-between;">
<article style="background-color: lightblue;"> <h2>О цветах</h2>
<p>Цветы бывают тысячи разных форм и цветовых комбинаций, каждый из которых
имеет свое название и классификацию. Существует более чем 350 тысяч видов цветущих
растений, поэтому обязательно найдется цветок, который подойдет каждому человеку. Помимо
эстетического наслаждения цветы могут приносить практическую пользу людям. Например,
цветы могут использоваться в фармацевтике для приготовления различных лекарств. Также
из многих полевых цветов пчелы собирают пыльцу из которой получается очень вкусный мед.
</p>
</article>
<div>
<aside style="float: right; background-color: pink">
<h4>Цветы и мед</h4>
<p>Цветы активно используются для производства меда. Нравятся пчелам люпин,
календула, мальва, фиалки, розы, циннии, подсолнухи, одуванчики, маттиола и другие.
</p></aside> <aside style="float: right; background-color: lightgreen">
<h4>Цветы в фармацевтике и косметологии</h4> <p>В качестве лекарственных
растений широко используются такие цветы как алоэ, ромашка, гвоздика, шафран, эхинацея и
многие другие. </p>
</aside></div>
</div></body>
```

Реализация данного кода представлена на рисунке 3.7.

В данном примере содержимое блока `article` содержит информацию о цветах. В то время как два блока `aside` содержат информацию косвенно связано с основным контентом — цветы в косметологии и цветы в производстве меда.

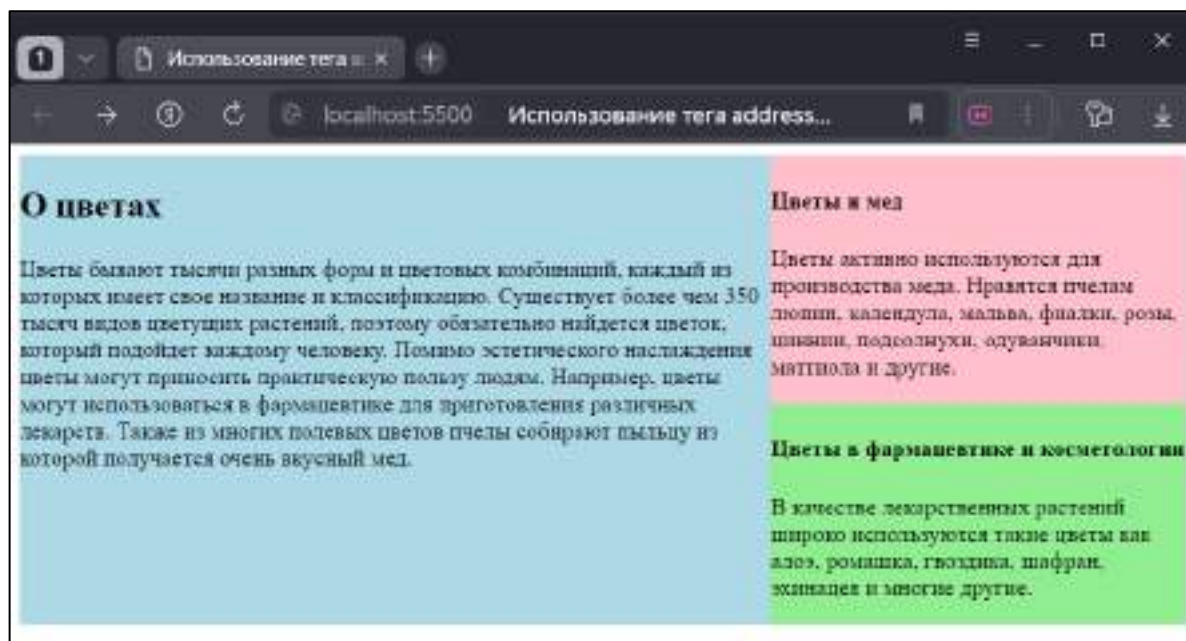


Рисунок 3.7 – Использование элемента aside в HTML5

3.8 Элемент main

Элемент main представляет основное содержимое веб-страницы. Он представляет уникальный контент, в который не следует включать повторяющиеся на разных веб-страницах элементы сайдбаров, навигационные ссылки, информацию о копирайте, логотипы и тому подобное.

```
<body> <main style="background-color: #F0E68C;"> <h1>Кошки</h1>
  <p>Кошка - домашнее животное, одно из наиболее популярных (наряду с собакой)
«животных-компаньонов». С точки зрения научной систематики, домашняя кошка -
млекопитающее семейства кошачьих отряда хищных. Одни исследователи рассматривают
домашнюю кошку как подвид дикой кошки, другие - как отдельный биологический вид.</p>
  <article><h2>Кошки в искусстве</h2>
  <p>Согласно сохранившимся до наших дней суевериям, кошки считаются
хранительницами домашнего очага и уюта. На новоселье принято первой впускать в дом
кошку. Из-за способности кошек приземляться на лапы при падении считается, что они
обладают особым «шестым чувством» и что у них девять жизней.</p> </main> </body>
```

Реализация данного кода представлена на рисунке 3.8.

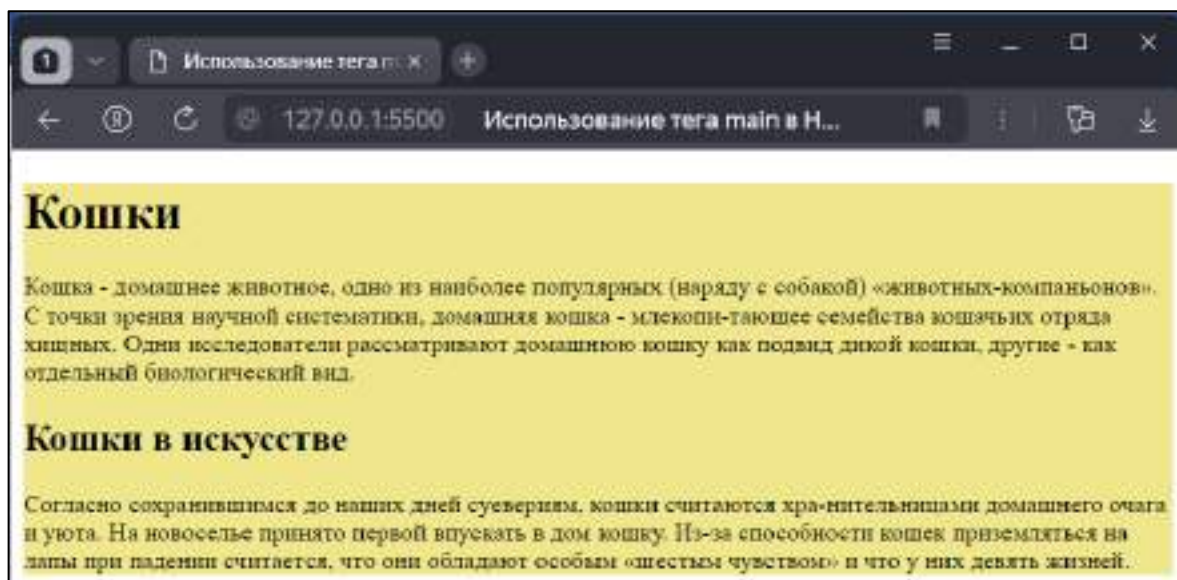


Рисунок 3.8 – Использование тега main в HTML5

3.9 Контрольные вопросы

- 1) Семантическая структура веб-страницы. Отличие элементов article и section.
- 2) Блок работы со ссылками. Элемент nav.
- 3) Заголовок, подвал и адресный блок в структуре веб-страницы, использование элементов header, footer и address соответственно.
- 4) Применение элемента aside в семантической верстке веб-страницы.
- 5) Особенности работы с элементом main.

3.10 Тестирование по главе

- 1) Элемент ... объединяет между собой части информации и выполняет их группировку
 - a) article;

- б) section;
- в) div;
- г) nav.

2) Nav, как правило, представляет из себя ...

- а) нумерованный список;
- б) ненумерованный список;
- в) нумерованный список с набором ссылок;
- г) ненумерованный список с набором ссылок.

3) Какой элемент может содержать заголовки, навигацию, формы поиска?

- а) header;
- б) footer;
- в) address;
- г) section.

4) Какой элемент обычно содержит даты публикации, блок ссылок на похожие ресурсы?

- а) header;
- б) footer;
- в) address;
- г) section.

5) Какой элемент предназначен для отображения контактной информации?

- а) header;
- б) footer;
- в) address;
- г) section.

б) Какой элемент должен быть идентифицирован с помощью включения в него заголовков?

- а) article;
- б) section;
- в) header;
- г) div.

7) Элемент ... призван содержать элементы навигации

- а) section;
- б) address;
- в) nav;
- г) aside.

8) Какой элемент можно использовать для сайдбаров, рекламных блоков?

- а) header;
- б) footer;
- в) aside;
- г) address.

9) Какой элемент представляет основное содержимое страницы?

- а) aside;
- б) body;
- в) main;
- г) div.

10) Наличие только одного элемента ... допустимо на странице

- а) header;
- б) footer;
- в) nav;
- г) main.

4 Работа с каскадными таблицами стилей

CSS – это формальный язык, служащий для описания оформления внешнего вида документа, созданного с использованием языка разметки (HTML, XHTML, XML). Название происходит от английского Cascading Style Sheets, что означает “каскадные таблицы стилей”.

Любой HTML-документ без использования стилей будет выглядеть лаконично. Каскадные таблицы стилей (CSS) определяют представление документа и его внешний вид. Назначение CSS – отделять то, что задает внешний вид страницы, от ее содержания. Если документ создан только с использованием HTML, то в нем определяется не только каждый элемент, но и способ его отображения (цвет, шрифт, положение блока и т. д.). Если же подключены каскадные таблицы стилей, то HTML описывает только очередность объектов. А за все их свойства отвечает CSS [5].

Стиль в CSS - это правило, которое сообщает веб-браузеру, как форматировать элемент. Форматирование может включать установку цвета фона элемента, установку цвета и типа шрифта и т. д. Определение стиля состоит из двух частей: селектора, указывающего на элемент, и блока объявления стиля, т.е. набора команд, устанавливающих правила форматирования. Например:

div {background-color:blue; width: 100px; height: 60px;}

В данном случае селектором является div. Этот селектор указывает, что этот стиль будет применен ко всем элементам div. После селектора находится блок объявления стиля, заключенный в фигурные скобки. Команды определяются между открывающими и закрывающими скобками. Каждая команда состоит из свойства и значения. Таким образом, в следующем выражении:

background-color:blue;

background-color представляет свойство, а blue - значение. Свойство определяет определенный стиль. Существует множество свойств CSS. Например, background-color определяет цвет фона. После двоеточия указывается значение этого свойства. Например, приведенная выше команда устанавливает для свойства background-color

значение красного цвета. Другими словами, цвет фона элемента устанавливается “blue”, то есть синий.

После каждой команды ставится точка с запятой, отделяющая данную команду от остальных. Наборы таких стилей часто называют таблицами стилей или CSS (Cascading Style Sheets или каскадные таблицы стилей). Существуют различные способы определения стилей.

4.1 Виды каскадных таблиц стилей

4.1.1 Внедренные таблицы стилей. Атрибут style

Первый способ заключается во встраивании стилей непосредственно в элемент с помощью атрибута style.

```
<body>
```

```
<h2 style="color: purple;"> Использование стилей </h2>
```

```
<div style="width: 200px; height: 200px; background-color: blue;">
```

```
</div> </body>
```

Реализация данного кода представлена на рисунке 4.1.

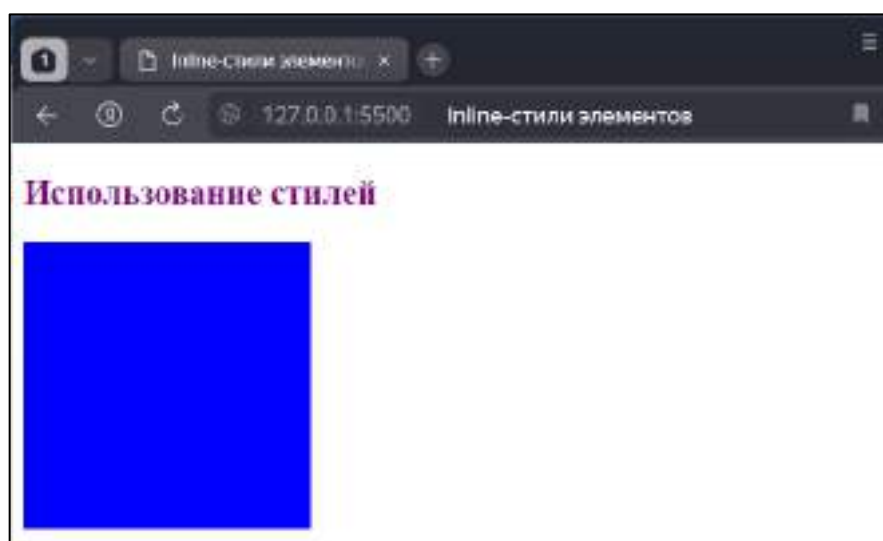


Рисунок 4.1 - Inline-стили элементов HTML5

Здесь определены два элемента: заголовок h2 и div. Заголовок имеет синий цвет текста, определенный с помощью свойства color. Блок div имеет свойства, определенные для ширины, высоты и цвета фона.

4.1.2 Внутренние таблицы стилей. Тег style

Второй способ - использовать элемент стиля в html-документе. Этот элемент сообщает браузеру, что данные внутри представляют собой код CSS, а не HTML:

```
<head>
  <title>Использование стилей</title>
  <style>  h2{color: purple; }
          div{width: 200px; height: 200px; background-color: blue; }
</style> </head>
  <body> <h2>Использование стилей</h2><div>
</div></body>
```

Результат в будет абсолютно такой же, как в предыдущем примере.

4.1.3 Внешние таблицы стилей. Файл *.css

Третий способ - перенесение стилей во внешний файл. Создается текстовый документ, переименовывается в style.css и определяется следующее содержимое:

```
h2 {color: purple;}
div {width: 200px; height: 200px; background-color: blue;}
```

Это те же стили, что были внутри элемента style. Исходный код html-страницы будет выглядеть следующим образом:

```
<head> <title>Использование стилей</title>
  <link rel="stylesheet" type="text/css" href="styles.css"/>
</head> <body> <h2>Использование стилей</h2> <div></div> </body>
```

Здесь больше нет элемента style, но есть элемент ссылки link, который связывает созданный выше файл style.css:

```
<link rel="stylesheet" type="text/css" href="styles.css"/>
```

Поэтому, определяя стили во внешнем файле, HTML-код становится чище, структура страницы отделяется от ее стилей. Стили, определенные таким образом, гораздо легче изменить, если бы они были определены внутри элементов или в элементе стиля. Возможно объединить все эти подходы, и для одного элемента некоторые свойства CSS определяются внутри самого элемента, другие свойства CSS определяются в элементе стиля, а другие находятся во внешнем файле.

Например:

```
<head>
  <link rel="stylesheet" type="text/css" href="styles.css"/>
  <style>
    div {width:100px;}
  </style> </head>
<body> <div style="width:200px;"></div></body>
```

А в файле styles.css определены стили:

```
div{ width:10px; height:300px; background-color:blue;}
```

В этом случае элемент div имеет свойство ширины, определенное в трех местах с разными значениями. К элементу будет применена следующая система приоритетов: если у элемента определены встроенные стили (inline-стили), то они имеют наивысший приоритет, т.е. в примере выше конечная ширина будет равна 200 пикселей. Следующими по приоритету являются стили, определенные в элементе style. Стили с самым низким приоритетом определены во внешнем файле.

4.2 Задание стилей с помощью селекторов

4.2.1 Селекторы типа

Определение стиля начинается с селектора. В CSS3 определено три вида селекторов – селектор типа, селектор класса и ID-селектор. Селектор типа определяет HTML-элемент, к которому будет применен стиль. Селекторы типа легко определить

в стилях, так как они имеют одноимённые наименования с форматируемыми элементами, например: `p` - сообщает браузеру, что необходимо отформатировать все HTML теги `<p>` (параграф).

```
P {font-family: 'Times New Roman', color: green;}
```

Задаются селекторы типа в заголовке html документа, в то время как переопределение стиля происходит в теле документа.

```
<head> <title>Использование селекторов типа</title>  
  <style> h2 { font-family: 'Courier New', Courier, monospace; } </style>  
</head> <body>  
  <h1>Селекторы в CSS (без стилей)</h1>  
  <h2>Селекторы в CSS (с использованием стилей)</h2> </body>
```

Реализация данного кода представлена на рисунке 4.2.

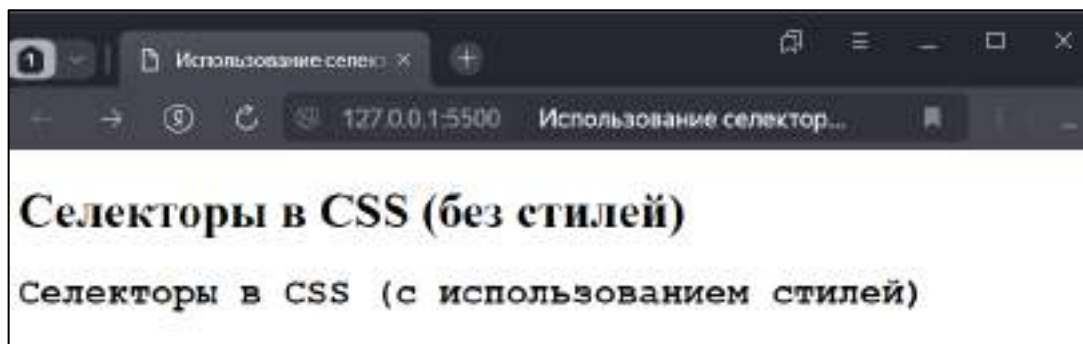


Рисунок 4.2 – Селекторы типа CSS

4.2.2 Селекторы класса

Иногда одни и те же элементы требуют разного оформления. В этом случае можно использовать классы. Чтобы определить селектор класса в CSS, следует поставить точку перед именем класса:

```
.greenBlock{background-color:green;}
```

Имя класса может быть любым. Например, в данном случае имя класса – “greenBlock”. Однако в имени класса могут использоваться буквы, цифры, дефисы и подчеркивания, и имя класса должно начинаться с буквы. Также стоит учитывать регистр зависимости имен: имена «article» и «ARTICLE» представляют разные классы.

После определения класса его можно применить к элементу с помощью атрибута class. Например:

```
<div class="greenBlock"></div>
```

Далее приведен пример использования селекторов класса.

```
<head> <style>
```

```
.blocks {width: 100vw; display: flex; justify-content: start;}
```

```
div {width: 250px;margin: 10px; padding: 10px;}
```

```
span {font-size: 20px;}
```

```
.greenBlock {background-color: green; color: white;}
```

```
.yellowBlock {background-color: yellow;}
```

```
.blueBlock {background-color: blue; color: white;}
```

```
</style> </head><body> <h2>Использование классов</h2>
```

```
<div class="blocks"><div class="greenBlock"> <h2>Зеленый цвет</h2> <span>  
<b>Значение.</b> Зелёный цвет способен гасить эмоции, успокаивать, создавать ощущение  
умиротворения и безопасности и потому воспринимается как один из самых  
комфортных.</span>
```

```
</div><div class="yellowBlock"> <h2>Жёлтый цвет</h2> <span> <b>Значение.</b> Это  
один из самых тёплых цветов видимого спектра. В первую очередь жёлтый ассоциируется с  
солнцем и в основном вызывает положительные эмоции. </span> </div>
```

```
<div class="blueBlock"> <h2>Синий цвет</h2> <span> <b>Значение.</b> Синий также  
воспринимается как успокаивающий, снижающий тревогу. Он создаёт приятные ассоциации  
с морем и небом. </span> </div> </div> </body>
```

Реализация данного кода представлена на рисунке 4.3.

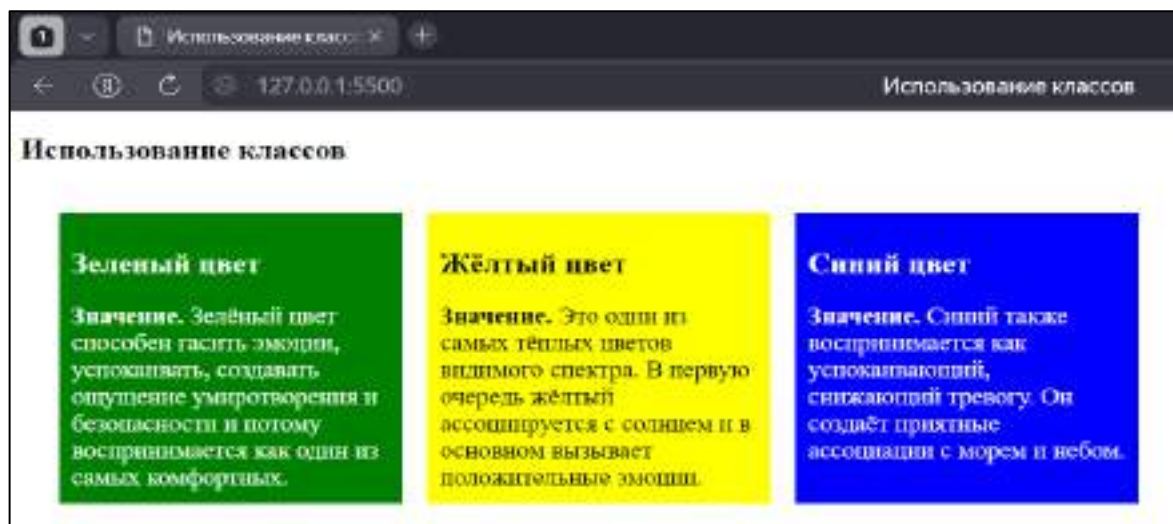


Рисунок 4.3 – Селекторы классов в CSS3

Другой пример использования селектора класса. Допустим, необходимо создать Web-страницу, на которой будет два вида абзацев <p>, причем оба вида будут постоянно чередоваться и часто повторяться.

Типичный пример такой страницы - интервью, в котором чередуются вопросы журналиста и ответы интервьюируемого. При создании такой страницы необходимо визуально отделить вопросы и ответы друг от друга. Это будет выглядеть следующим образом:

```
<head> <style> body { background-color: rgba(247, 255, 171, 0.449); }
    .ask { font-family: Arial, sans-serif;
    font-style: italic; font-weight: bold;
    font-size: 14px; margin-left: 15px; }
    .answer, ul {font-family: Courier, serif;
    font-size: 14px; margin-left: 10px;}
</style> </head> <body>
<h3>Экзамен по дисциплине “Основы веб-разработки”</h3>
<p class="ask">1. Что такое CSS?</p>
<p class="answer"> CSS - это язык стилизации, который делает веб-страницы HTML
более презентабельными. Он позволяет, среди прочего, добавлять на веб-сайт цвет,
    дизайн и кнопки.</p>
<p class="ask">2. Каковы преимущества использования CSS?</p>
<p class="answer"> Использование CSS имеет множество преимуществ, в том числе:
<ul> <li>Специальные возможности</li> <li>Переформатирование страницы</li>
    <li>Согласованность для всего сайта</li> <li>Пропускная способность</li>
    <li>Разделение полосы пропускания презентации</li>
</ul> <p class="ask">3. Каковы недостатки CSS?</p>
<p class="answer">Недостатками использования CSS являются:
<ul> <li>Тексты, оформление и правила, ориентированные на конкретную аудито-
рию, не допускаются</li>
    <li>Нет выражений</li>
    <li>Отсутствует динамическое поведение для управления псевдоклассом</li>
    <li>Невозможно подняться по селекторам</li>
    <li>Вертикальный контроль ограничен</li>
    <li>Объявление столбца отсутствует</li> </ul> </body>
```

Реализация данного кода представлена на рисунке 4.4.

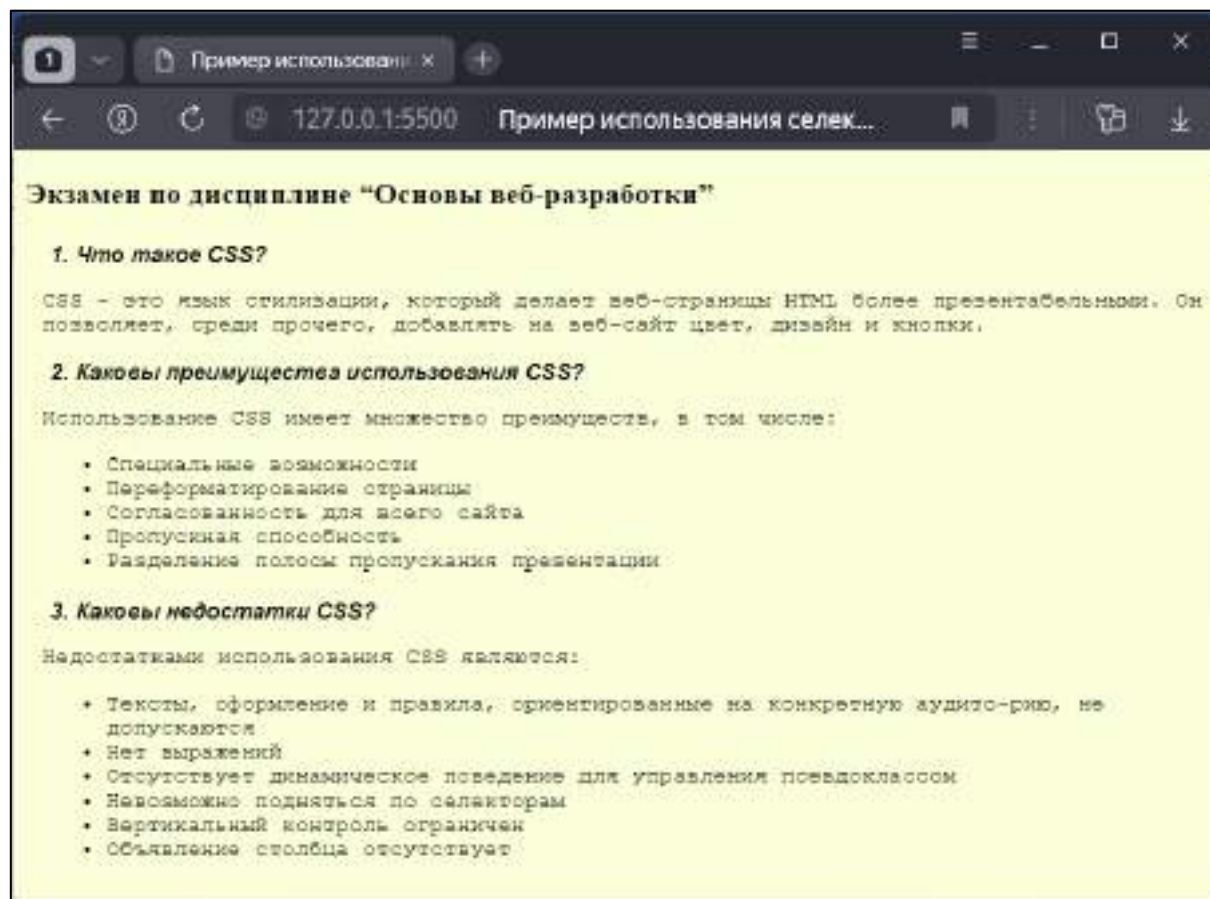


Рисунок 4.4 - Использование селектора класса

4.2.3 Id-селектор

Для идентификации уникальных на веб-станции элементов используются идентификаторы, которые определяются с помощью атрибутов id. Определение стилей для идентификаторов аналогично определению классов, только вместо точки ставится символ решетки #.

Например,

```
<head> <style>  
#mystyle {text-color: aqua;}  
</style></head>
```

Вызов стилей реализуется в теле документа в теге <body>.

```
<h1 id="mystyle"></div>
```

Далее представлен пример использования id-селекторов при задании стилей для тега <div>.

```
<head> <style> div{ margin: 10px;
        border: 2px solid #000000;}
        #header{ height: 65px; background-color: #ffE87C;}
        #content{ height: 95px; background-color: gold;}
        #footer{ height: 25px; background-color: #FFF380;}
    </style></head>
```

```
<body> <div id="header"> <h2>Золотая рыбка</h2> </div>
```

<div id="content"> Золотая рыбка (Carassius auratus) это по праву самый узнаваемый вид декоративных рыб, который занимает почетное первое место в списке наиболее популярных аквариумных жителей. Свое название золотая рыбка получила, прежде всего, за цвет чешуи: стандартный цвет тела – красновато-золотистый, зачастую с металлическим отливом. </div>

```
    <div id="footer">Copyright &copy; Golgfish 2024 </div>
```

```
</body>
```

Реализация данного кода представлена на рисунке 4.5.

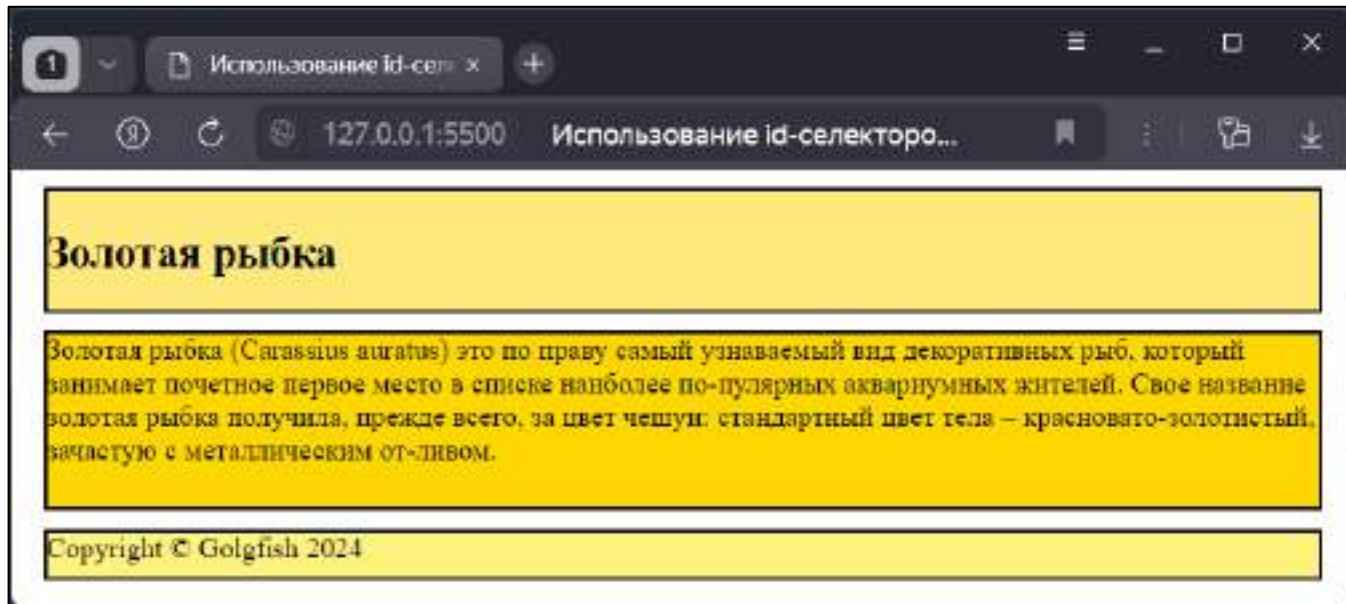


Рисунок 4.5 – Использование id-селекторов в CSS3

Однако стоит заметить, что идентификаторы в большей степени относятся к структуре веб-странице и в меньшей степени к стилизации. Для стилизации преимущественно используются классы, нежели идентификаторы.

4.2.4 Универсальный селектор

Помимо селекторов тегов, классов и идентификаторов, CSS также имеет универсальный селектор, который представляет собой знак звездочки (*). Он применяет стили ко всем элементам HTML-страницы:

```
*{ background-color: #00FA9A;}
```

Далее представлен пример использования универсальных селекторов.

```
<head> <style>
```

```
  * { background-color: #FF69B4; }
```

```
    div { margin: 15px;border: 2px solid #000000; }
```

```
</style> </head> <body>
```

```
  <div>Оренбургский государственный университет - ОГУ</div>
```

```
  <div>История Оренбургского государственного университета началась 14 сентября  
1955 года, когда в городе Чкалове (ныне Оренбурге) было открыто вечернее отделение  
Куйбышевского индустриального института им. В. В. Куйбышева</div>
```

```
  <div> Copyright &copy; 2024</div> </body>
```

Реализация данного кода представлена на рисунке 4.6.

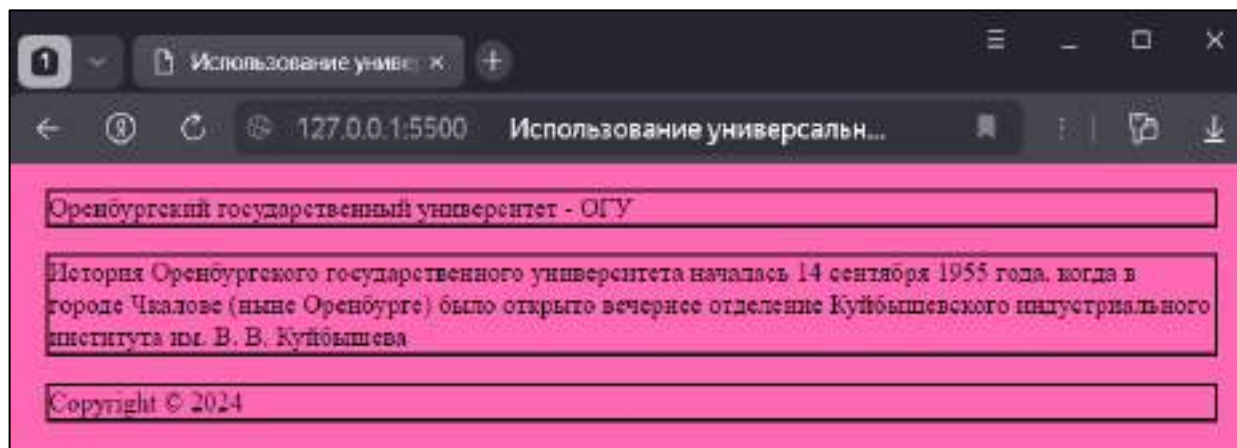


Рисунок 4.6 – Использование универсальных селекторов

4.2.5 Стилизация группы селекторов

Иногда к набору селекторов применяются определенные стили. Например, необходимо применить подчеркивание ко всем заголовкам. В этом случае можно перечислить селекторы всех элементов через запятую [1, 16].

```

<head> <style> h1, h2, h3 {color: crimson;} </style> </head>
<body>
<h1>Актёры</h1>
<h2>Никита Кологривый</h2>
<h3>Биография</h3>
<p>Никита Сергеевич Кологривый - российский актёр театра и кино.
Родился 16 октября 1994 года в Новосибирске. В юности занимался боксом и
планировал поступать в педагогический университет на тренера, но поступил в
Новосибирский театральный институт. После окончания второго курса отправился в Москву,
где поступил в ГИТИС на курс театральных режиссёров Сергея Голомазова и Павла Хомского.
С отличием окончил вуз в 2018 году.</p>
</body>

```

Реализация данного кода представлена на рисунке 4.7.

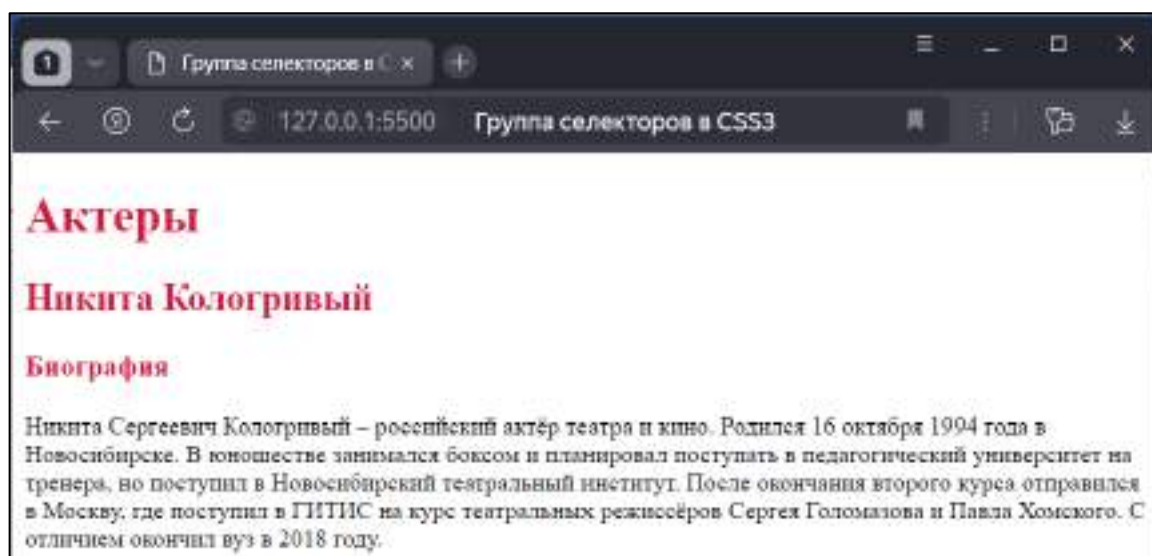


Рисунок 4.7 - Группа селекторов в CSS3

Группа селекторов может содержать как селекторы тегов, так и селекторы классов и идентификаторов, например:

```
h1, #header, .blueBlock {color: blue;}
```

4.3 Селекторы потомков

Веб-страница может иметь сложную организацию, одни элементы внутри себя могут определять другие элементы. Вложенные элементы иначе можно назвать потомками. А контейнер этих элементов - родителем.

Например, пусть элемент `body` на веб-странице имеет следующее содержимое:

```
<body>
<h2>Название</h2>
  <div> <p>Контент</p> </div>
</body>
```

Внутри элемента `body` определено три вложенных элемента: `h2`, `div`, `p`. Все эти элементы являются потомками элемента `body`. А внутри элемента `div` определен только один вложенный элемент - `p`, поэтому элемент `div` имеет только одного потомка. Используя специальные селекторы, можно стилизовать вложенные элементы или потомков внутри строго определенных элементов.

Например, на странице могут быть параграфы внутри блока с основным содержимым и внутри блока футера. Но для параграфов внутри блока основного содержимого можно установить один шрифт, а для параграфов футера другой.

```
<head>
<style> #main p { font-size: 17px; background-color: mediumspringgreen;}
  #footer p { background-color: mediumseagreen; color: white;}
</style> </head>
<body> <div id="main"><p>Интересные факты о цветах</p>
  <p> По факту, если мы видим бутон белого цвета, значит, его лепестки совершенно
бесцветные и свободно отражают белый солнечный свет. Своего оттенка у них нет. Цветов
чистого черного цвета тоже не бывает: скорее всего, они темно-лиловые, фиолетовые, темно-
бордовые. Больше у цветов нет никаких ограничений по оттенкам: они могут быть абсолютно
любыми, кроме черного и белого. </p> </div>
  <div id="footer"><p> Интересно, что пчелы никогда не ошибаются в выборе цветка:
сажаются только на те, у которых есть нектар. Энтомологи выяснили: это происходит потому,
что на лепестках цветов есть невидимые линии и пятна, видимые только в ультрафиолете. По
этим светящимся узорам насекомые ориентируются по цветам. </p>
</div> </body>
```

Реализация данного кода представлена на рисунке 4.8.

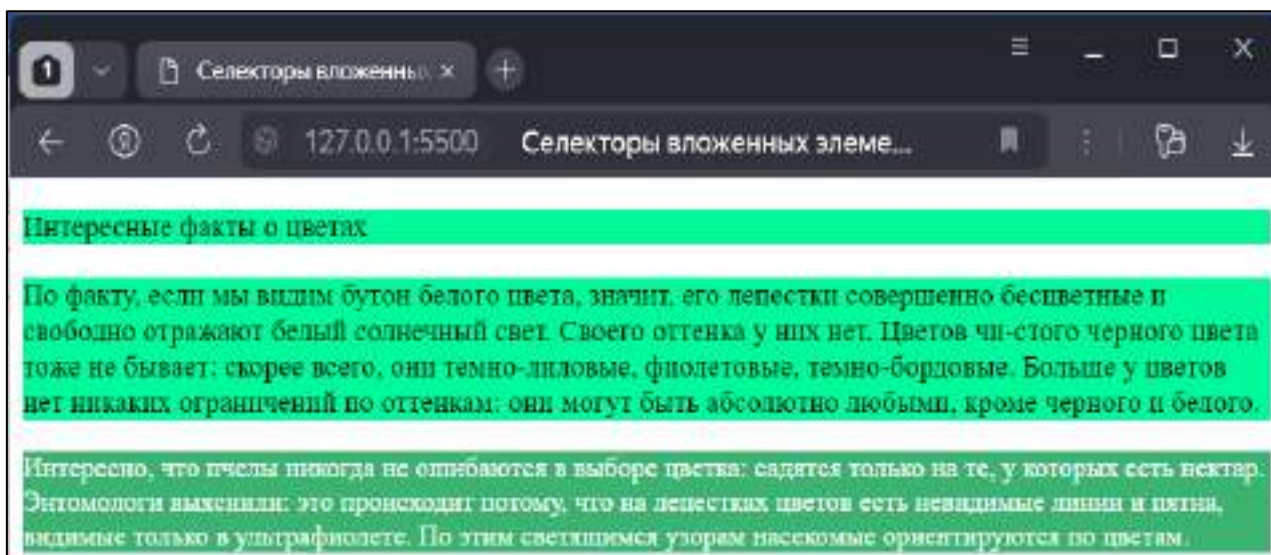


Рисунок 4.8 - Селекторы вложенных элементов в CSS3

Далее представлен еще один пример работы с дочерними селекторами.

```
<head>
<style>
li .blueLink {color: blue;}</style></head>
<body> <h2>Жанры книг</h2> <ul>
    <li>Жанр книги “Призрак Оперы”: <a class="blueLink">Мистика</a></li>
    <li>Жанр книги “Безмолвный пациент”: <a>Ужасы</a></li>
    <li>Жанр книги “Судьба человека”: <a>Драма</a></li>
    <li>Жанр книги “Укрощение строптивой”: <a>Комедия</a></li>
</ul></body>
```

Реализация данного кода представлена на рисунке 4.9.

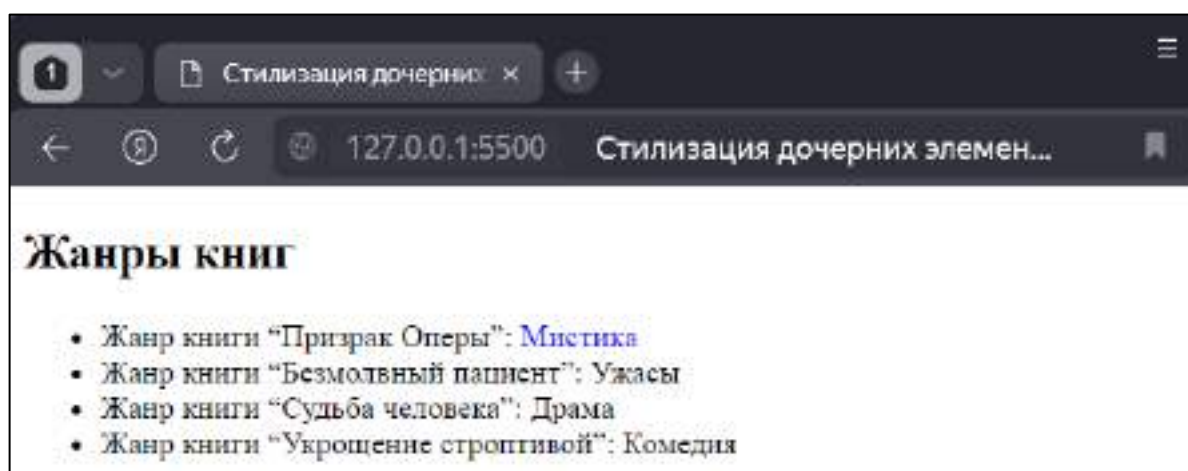


Рисунок 4.9 - Стилизация дочерних элементов в CSS3

Здесь стиль применяется к элементам с классом "blueLink", которые находятся внутри элемента . И соответственно браузер окрасит эти элементы в синий цвет:

Пробел: li .blueLink играет большое значение и указывает как раз, что элементы с классом blueLink должны быть вложенными по отношению к элементу .

Но если убрать пробел:

li.blueLink {color: blue;}

то смысл селектора изменится. Теперь будет подразумеваться, что стиль применяется к элементам , которые имеют класс blueLink. Например, к следующему элементу:

**<li class="blueLink"> Жанр книги “Безмолвный пациент”: <a> Ужасы **

Но никак не к элементу:

Жанр книги “Призрак Оперы”: Мистика

4.4 Селекторы дочерних элементов

Селекторы дочерних элементов отличаются от селекторов потомков тем, что позволяют выбрать элементы только первого уровня вложенности. Например:

**<body>
 <h2>Название</h2><div> <p>Контент</p> </div>
</body>**

Хотя вложенными в элемент body элементами являются целых три - h2, div, p, но дочерними из них являются только два - div и h2, так как они находятся в первом уровне вложенности. А элемент p находится на втором уровне вложенности, так как вложен внутрь элемента div, а не просто элемента body.

Для обращения к дочерним элементам используется знак угловой скобки. Далее представлен пример работы с дочерними элементами.

<head> <style> .article > p {font-weight: bold; color: steelblue; border: 2 px;} </style>


```
</head> <body> <div class="article">
```

```
  <p>Хасеки Хюрем Султан</p>
```

```
  <div class="content">
```

```
    <p> Хюрем - наложница, а затем супруга османского султана Сулеймана I  
    Великолепного, мать султана Селима II. Являлась одной из самых влиятельных женщин в  
    истории Османской империи. </p> <p> Во время одного из набегов крымских татар девушка  
    попала в плен и после нескольких перепродаж, была подарена Сулейману по случаю  
    восшествия на престол.</p> </div> </div> </body>
```

Реализация данного кода представлена на рисунке 4.10.

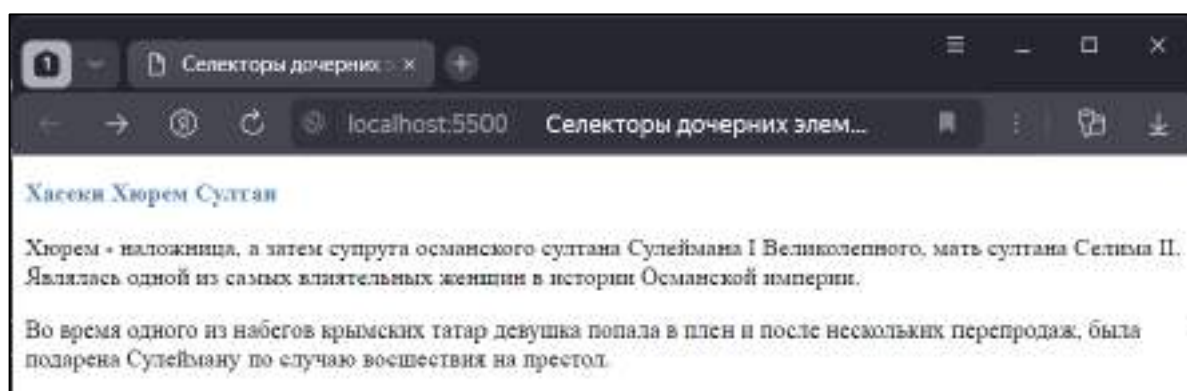


Рисунок 4.10 - Селекторы дочерних элементов в CSS3

В блоке с классом article есть два параграфа. Селектор `.article > p` выбирает только те параграфы, который находятся непосредственно в блоке article.

Если бы использовался другой селектор без символа `>`

```
.article p {font-weight: bold; color: steelblue; border: 2 px;}
```

Тогда стиль бы применялся ко всем параграфам на всех уровнях вложенности (рисунок 4.11).

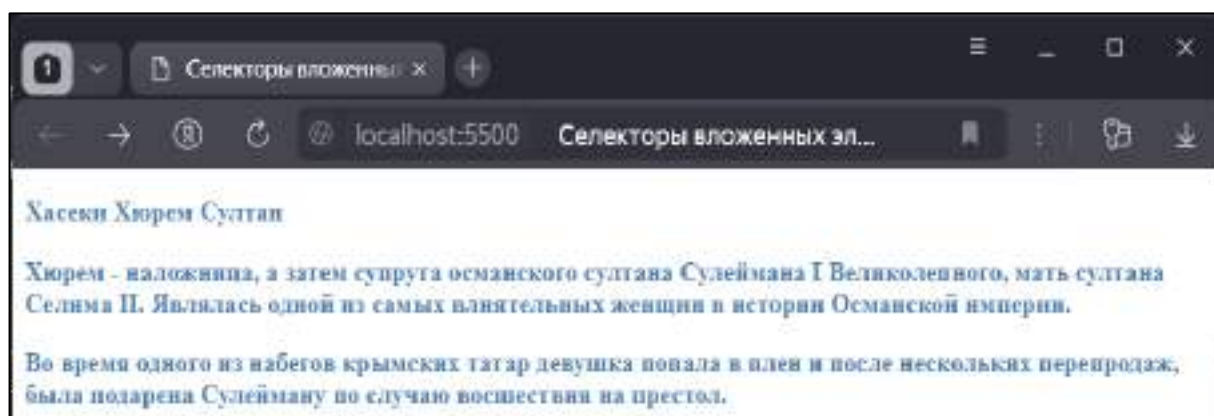


Рисунок 4.11 - Селекторы вложенных элементов в CSS

4.5 Селекторы элементов одного уровня

Селекторы элементов одного уровня или смежных элементов позволяют выбрать элементы, которые находятся на одном уровне вложенности. Иногда такие элементы еще называют сиблинги (siblings) или сестринскими элементами.

```
<body>
  <h2>Заголовок</h2>
  <div> <p>Текст первого блока</p></div>
  <div> <p>Текст второго блока</p></div>
</body>
```

Здесь элементы h2 и оба блока div являются смежными, так как находятся на одном уровне. А элементы параграфов и заголовков h2 не являются смежными, так как параграфы вложены в блоки div.

Чтобы стилизовать первый смежный элемент по отношению к определенному элементу, используется знак плюса +.

Далее представлен пример работы с селекторами одного уровня.

```
<head>
  <style> h2+div {color: LimeGreen; font-style: italic;} </style> </head>
  <body>
    <h2>Гарик Харламов</h2>
    <div> <p> Родился 28 февраля 1980 года в Москве. Отец Гарика после развода
переехал в Чикаго. Сын по окончании школы отправился за ним и получал образование в
знаменитой актерской школе. Через 5 лет молодой человек вернулся в Москву. Окончил
Государственный университет управления по специальности “Управление персоналом”.
</p></div>
    <div> <p> 7 лет играл в КВН за команды “Сборная Москвы” и “Незолотая молодёжь”.
Участник юмористического шоу “Comedy Club” (ТНТ), выступает в дуэте с Тимуром
Батрутдиновым. В отдельных выпусках Гарик выходит на сцену не под настоящим именем,
а под псевдонимом Эдуард Суровый.</p>
  </div> </body>
```

Реализация данного кода представлен на рисунке 4.12.

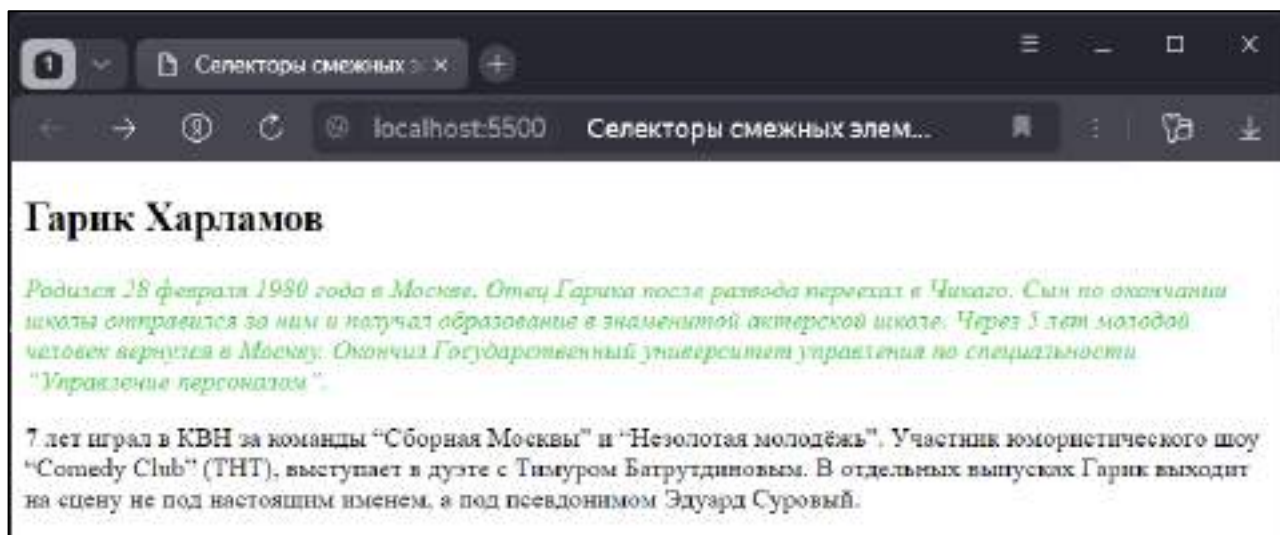


Рисунок 4.12 - Селекторы смежных элементов в CSS3

Селектор `h2+div` позволяет определить стиль (в данном случае красный цвет текста) для блока `div`, который идет непосредственно после заголовка `h2`. Причем этот селектор будет стилизовать блок `div`, если он будет идти непосредственно после заголовка. Если же между заголовком и блоком `div` будет находиться еще какой-либо элемент, то к нему не будет применяться стиль, например:

```
<h2>Заголовок</h2>
```

```
<p>Элемент между заголовком и блоком div</p>
```

```
<div> <p>Текст первого блока</p> </div>
```

Если надо стилизовать вообще все смежные элементы одного уровня, неважно непосредственно идут они после определенного элемента или нет, то в этом случае вместо знака плюса необходимо использовать знак тильды “~”.

```
<head>
```

```
<style> h2 ~ div {color: BlueViolet; font-weight: bold;} </style> </head>
```

```
<body> <h2>Иван Филиппович Янковский</h2>
```

```
<div> <p> Иван Филиппович Янковский - российский актёр театра и кино.
Пятикратный лауреат премии “Золотой орёл”: дважды за “Лучшую мужскую роль в кино”,
дважды за “Лучшую мужскую роль второго плана” и один раз за “Лучшего актёра онлайн-
сериала”. </p></div>
```

```
<div> <p> Янковский женат, его спутница - коллега по цеху Диана Пожарская. В 2021
году стало известно, что у них родился сын Олег. До супружества Янковский встречался с
```

дочерью известного бизнесмена Камиллой Байсаровой, а также был замечен в отношениях с дочерью лидера группы “Алиса” Верой Панфиловой. </p> </div> </body>

Реализация данного кода представлено на рисунке 4.13.

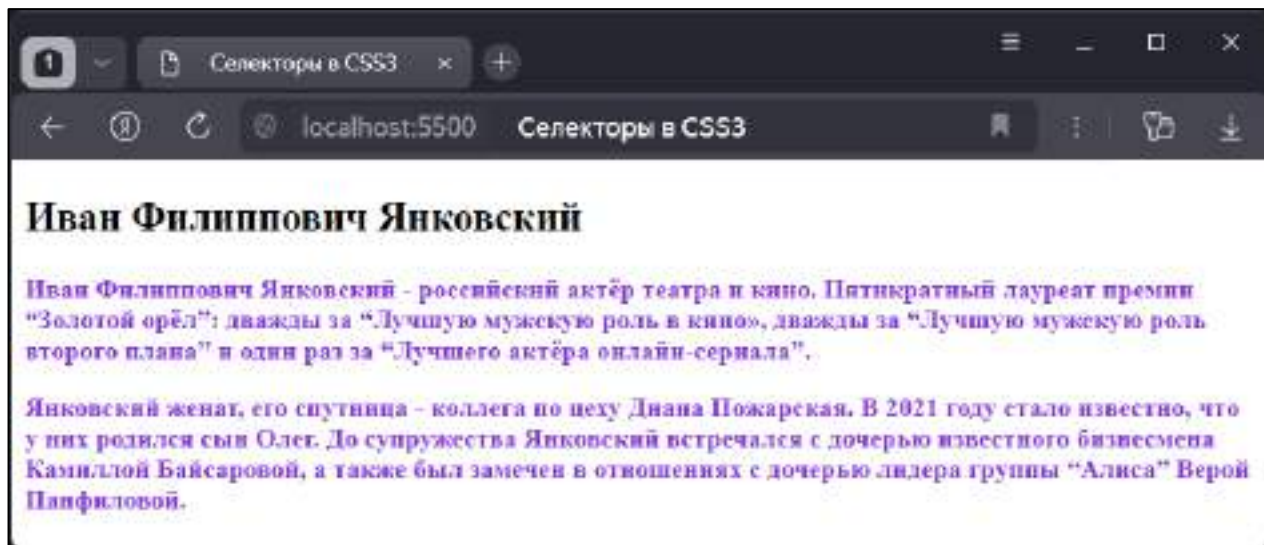


Рисунок 4.13 - Стилизация элементов одного уровня в CSS

4.6 Псевдоклассы

В дополнение к селекторам тегов, классов и идентификаторов доступны селекторы псевдоклассов, которые несут дополнительные возможности по выбору нужных элементов. Список доступных псевдоклассов:

- :root: позволяет выбрать корневой элемент веб-страницы;
- :link: применяется к ссылкам и представляет ссылку в обычном состоянии, по которой еще не совершен переход;
- :visited: применяется к ссылкам и представляет ссылку, по которой совершен переход;
- :active: применяется к ссылкам и представляет ссылку в тот момент, когда пользователь осуществляет по ней переход;
- :hover: представляет элемент, на который пользователь навел указатель мыши;
- :focus: представляет элемент, который получает фокус;

– :not: позволяет исключить элементы из списка элементов, к которым применяется стиль;

– :lang: стилизует элементы на основании значения атрибута lang;

– :empty: выбирает элементы, которые не имеют вложенных элементов, то есть являются пустыми.

При применении псевдоклассов перед ними всегда ставится двоеточие. Далее представлен пример, в котором стилизуются ссылки, используя псевдоклассы.

```
<head>
<title>Использование псевдоклассов</title> <style>
  a:link {font-size: 20px; color: ForestGreen; font-style: italic; }
  a:visited {font-size: 12 px; color: FireBrick;}
  a:hover {font-size: 24px; color: DarkViolet; text-decoration: underline;}
  a:active {font-size: 18px; color: DodgerBlue; свойство font-weight: bold;}
</style></head>
<body> <h2>Моя домашняя страница</h2>
<h3>Добро пожаловать на мою домашнюю страницу. Здесь я поделюсь самым
сокровенным. Расскажу про свои увлечения, питомцев, друзей и многое другое. Переходи по
ссылкам, будет интересно. </h3>
  <a href="https://family.html/">Моя семья</a><br />
  <a href="https://friends.html/">Мои друзья </a><br />
  <a href="https://pets.html/">Мои животные</a><br />
  <a href="https://hobbi.html/">Мое хобби</a><br /> </body>
```

Реализация данного кода представлена на рисунке 4.14.

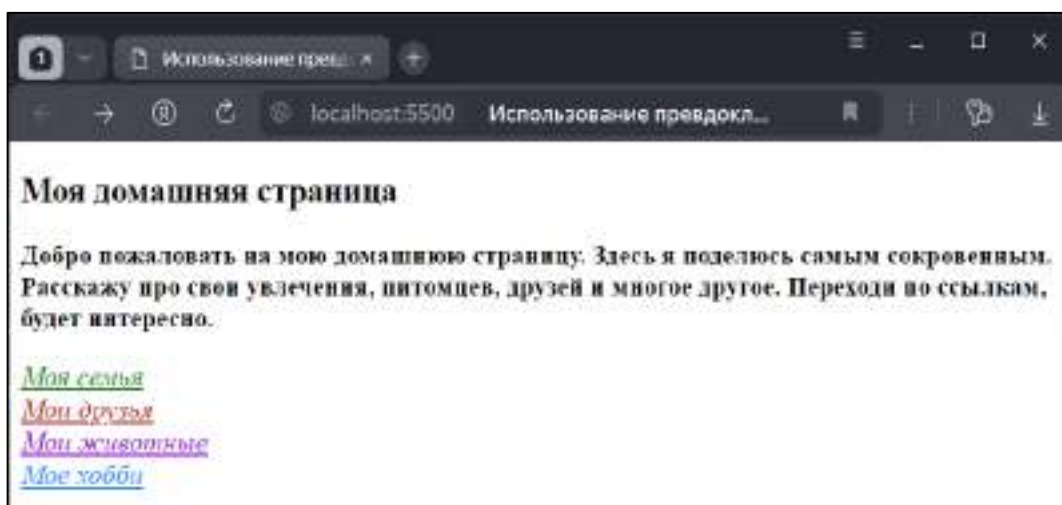


Рисунок 4.14 – Пример стилизации ссылок

Селектор `:lang` выбирает элементы на основании атрибута `lang`. Далее приведен пример работы с псевдоклассом `lang`.

```
<head> <style>  :lang(ru) { color: blue;}  
  </style> </head> <body> <form>  
<p lang="ru">Моя любимая актриса Марина Александрова </p>  
<p lang="en">My favorite actress is Marina Alexandrova</p>  
<p lang="de">Meine Lieblingsschauspielerin ist Marina Alexandrova</p>  </form></body>
```

Реализация данного кода представлена на рисунке 4.15.

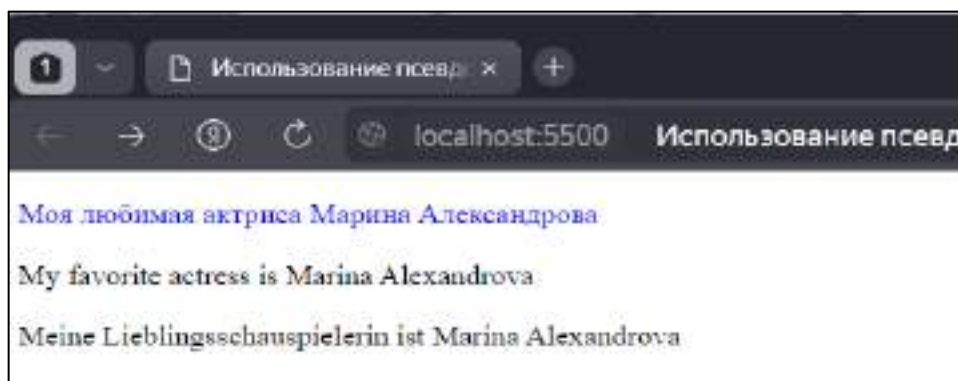


Рисунок 4.15 – Использование псевдокласса `:lang`

4.7 Псевдоклассы дочерних элементов

Особую группу псевдоклассов образуют псевдоклассы, которые позволяют выбрать определенные дочерние элементы [17]:

- `:first-child`: первый дочерний элемент;
- `:last-child`: последний дочерний элемент;
- `:only-child`: единственный дочерний элемент в некотором контейнере;
- `:only-of-type`: единственный элемент определенного типа (тега) в некотором контейнере;
- `:nth-child(n)`: дочерний элемент, который имеет определенный номер `n`;
- `:nth-last-child(n)`: дочерний элемент, который имеет определенный номер `n`, начиная с конца;

– :nth-of-type(n): дочерний элемент определенного типа, который имеет определенный номер;

– :nth-last-of-type(n): дочерний элемент определенного типа, который имеет определенный номер, начиная с конца.

4.7.1 Псевдокласс first-child

Далее приведен пример использования псевдокласса first-child для выбора первых ссылок в блоках.

```
<head> <style> a:first-child {font-weight:bold; color: BlueViolet;}  
    </style> </head>  
<body> <h3>Воздушный транспорт</h3>  
    <div> <a>самолёт</a><br/>  
        <a>геликоптер (вертолёт)</a><br/>  
        <a>планер</a><br/>  
        <a>аэростат и дирижабль</a><br/> </div>  
<h3>Наземный транспорт</h3> <div> <p>городской транспорт:</p>  
    <a>автомобиль</a><br/>  
    <a>трамвай</a><br/>  
    <a>автобус</a><br/>  
    <a>троллейбус</a> </div> </body>
```

Реализация данного кода представлена на рисунке 4.16.

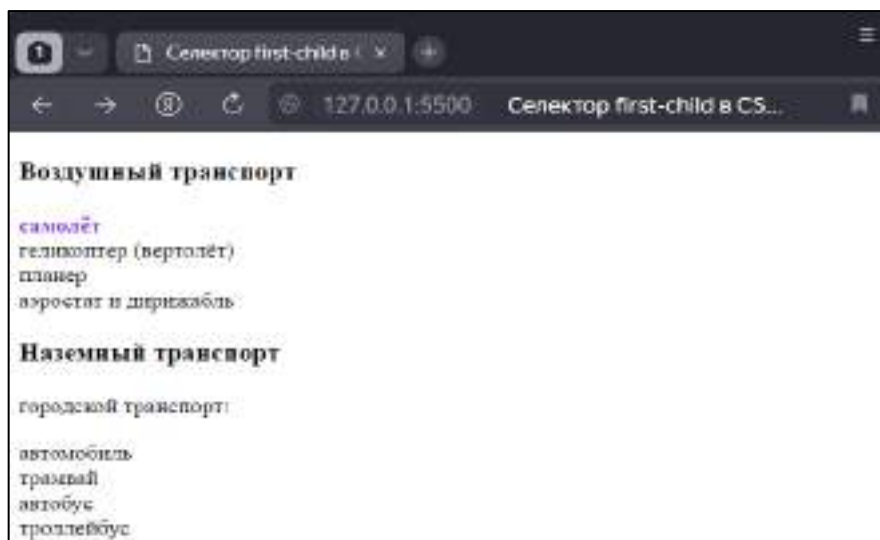


Рисунок 4.16 - Селектор first-child в CSS 3

Стиль по селектору `a:first-child` применяется к ссылке, если она является первым дочерним элементом любого элемента. В первом блоке элемент ссылки является первым дочерним элементом, поэтому к нему применяется заданный стиль. Во втором блоке первым элементом является параграф `<p>`, к нему не применяется стиль.

4.7.2 Псевдокласс `last-child`

Далее представлено использование псевдокласса `last-child`.

```
<head> <style>
    a:last-child{font-weight:bold; color:seagreen;} </style> </head>
<body> <h3>Воздушный транспорт</h3>
    <div> <a>самолёт</a><br/>
        <a>геликоптер (вертолёт)</a>
<br/> <a>планер</a><br/>
        <a>аэростат и дирижабль</a> </div>
<h3>Наземный транспорт</h3>
<div> <a>автомобиль</a><br/>
    <a>трамвай</a><br/>
    <a>автобус</a><br/>
    <a>троллейбус</a>
<p> - городской транспорт</p> </div> </body>
```

Реализация данного кода представлена на рисунке 4.17.

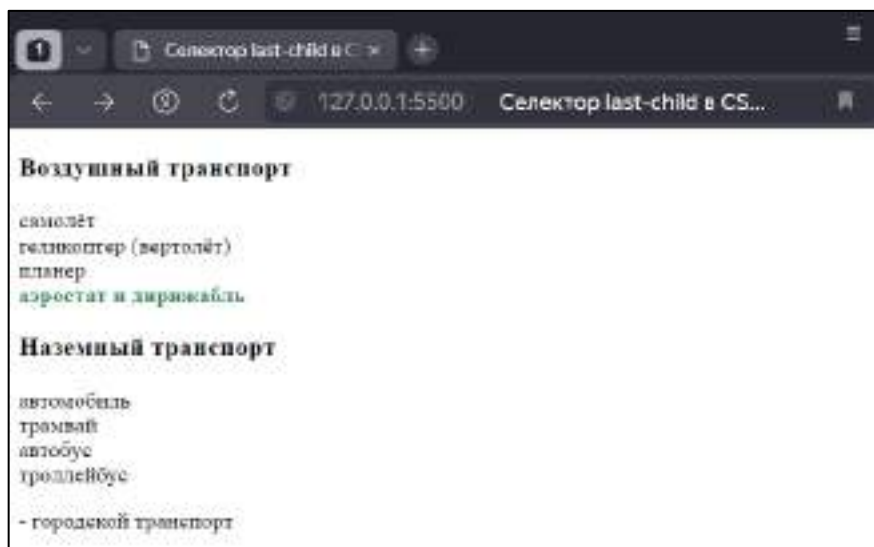


Рисунок 4.17 - Селектор `last-child` в CSS 3

Селектор `a:last-child` определяет стиль для ссылок, которые являются последними дочерними элементами.

В первом блоке как раз последним дочерним элементом является ссылка. А вот во втором последним дочерним элементом является параграф, поэтому во втором блоке стиль не применяется ни к одной из ссылок.

4.7.3 Селектор `only-child`

Селектор `:only-child` выбирает элементы, которые являются единственными дочерними элементами в контейнерах.

```
<head> <style> p:only-child{color: #8A2BE2; }  
    </style> </head>
```

```
<body> <h2>Александр Петров</h2>
```

```
    <div><p>Александр Петров, несмотря на молодой возраст, давно вошел в число  
самых востребованных российских актеров.</p></div>
```

```
    <div><p>Он активно снимается в кино, играет в театре, пишет стихи и пробует  
свои силы в режиссуре. </p> <p>Александр родился 25 января 1989 г. Родиной артиста  
является городок Переславль-Залесский в Ярославской области.</p> </div>
```

```
    <div> <p>Особый успех Александру Петрову принес сериал «Полицейский с Рублёвки»,  
который с 2016 года выходил на канале ТНТ. </p> </div> </body>
```

Реализация данного кода представлена на рисунке 4.18.

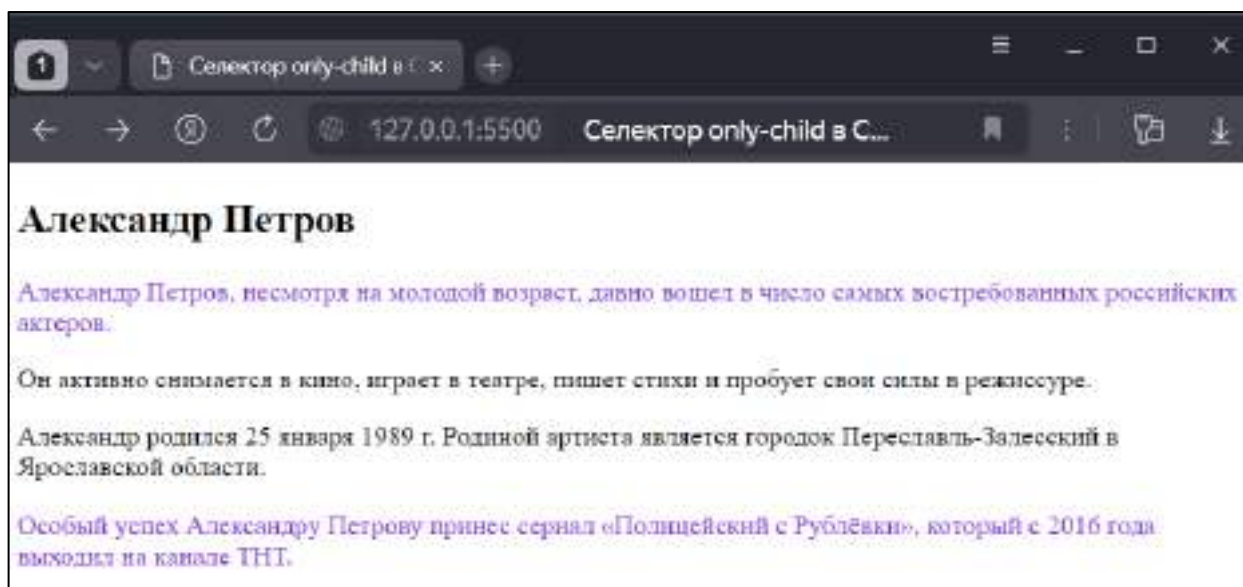


Рисунок 4.18 - Селектор `only-child` в CSS3

В данном случае первый и последний параграфы с текстами являются единственными дочерними элементами в своих внешних контейнерах, поэтому к ним применяется стиль - фиолетовый цвет шрифта.

4.8 Псевдоклассы форм

Ряд псевдоклассов используется для работы с элементами форм [18]:

- :enabled: выбирает элемент, если он доступен для выбора (то есть у него не установлен атрибут disabled);
- :disabled: выбирает элемент, если он не доступен для выбора (то есть у него установлен атрибут disabled);
- :checked: выбирает элемент, если у него установлен атрибут checked;
- :default: выбирает элементы по умолчанию;
- :valid: выбирает элемент, если его значение проходит валидацию HTML5;
- :invalid: выбирает элемент, если его значение не проходит валидацию;
- :in-range: выбирает элемент, если его значение находится в определенном диапазоне (для элементов типа ползунка);
- :out-of-range: выбирает элемент, если его значение не находится в определенном диапазоне;
- :required: выбирает элемент, если у него установлен атрибут required;
- :optional: выбирает элемент, если у него не установлен атрибут required.

4.8.1 Псевдоклассы enabled и disabled

Псевдоклассы enabled и disabled выбирают элементы форм в зависимости от того, установлен ли у них атрибут disabled.

Далее приведен приме использования псевдоклассов enabled и disabled.

<head> <style>

```

:enabled {border: 2px black solid; color: rgb(11, 184, 103); } </style> </head>
<body>
<h2>Регистрация пассажиров:</h2>
<h3>Введите свои данные</h3>
<p> Фамилия <input type="text" value="Обязательное поле для заполнения" /></p>
<p> Имя <input type="text" value="Обязательное для заполнения поле" /></p>
<p>Отчество<input type="text" disabled value="Необязательное поле" /></p> </body>

```

Реализация данного кода представлена на рисунке 4.19.

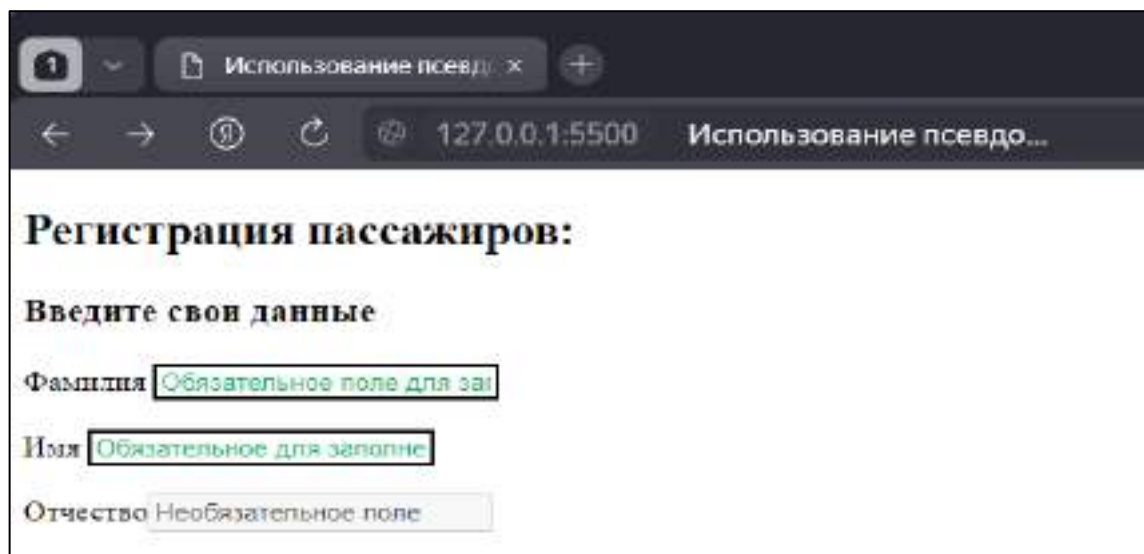


Рисунок 4.19 – Использование псевдоклассов enabled и disabled в CSS3

4.8.2 Псевдокласс checked

Псевдокласс checked стилизует элементы формы, у которых установлен атрибут checked. Далее представлен пример работы с этим псевдоклассом. В данном случае селектор :checked + span позволяет выбрать элемент, соседний с отмеченным элементом формы.

```

<head> <style>
:checked + span {color: SteelBlue; font-weight: bold; } </style> </head>
<body>
<h2>Туристическая фирма</h2>
<h3>Выберите курорт для отдыха:</h3>
<p> <input type="checkbox" checked/><span>Турция</span> </p>
<p> <input type="checkbox"/><span>Греция</span></p>

```

```

<p> <input type="checkbox"/><span>Кипр</span> </p>
<h3>Выберите класс отеля:</h3>
    <p> <input type="radio" checked /><span>Пять звезд</span>
    </p><p><input type="radio" /><span>Четыре звезды</span> </p>
</body>

```

Реализация данного кода представлена на рисунке 4.20.

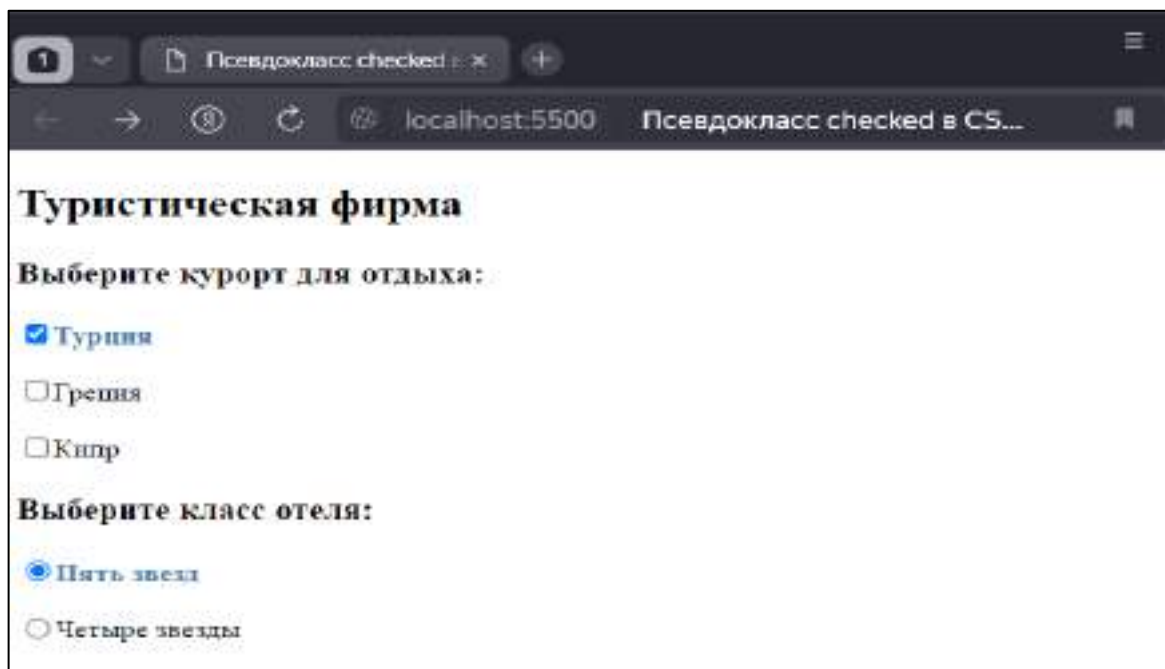


Рисунок 4.20 - Псевдокласс checked в CSS 3

4.8.3 Псевдокласс default

Псевдокласс :default выбирает стандартный элемент на форме. Как правило, в роли такого элемента выступает кнопка отправки.

```

<head>
<style>
:default { border: 2px solid rgb(76, 0, 255);}</style> </head>
<body>
<form>
<input name="login"/>
    <input type="submit" value="Авторизироваться" />
</form> </body>

```

Реализация данного кода представлена на рисунке. 4.21.

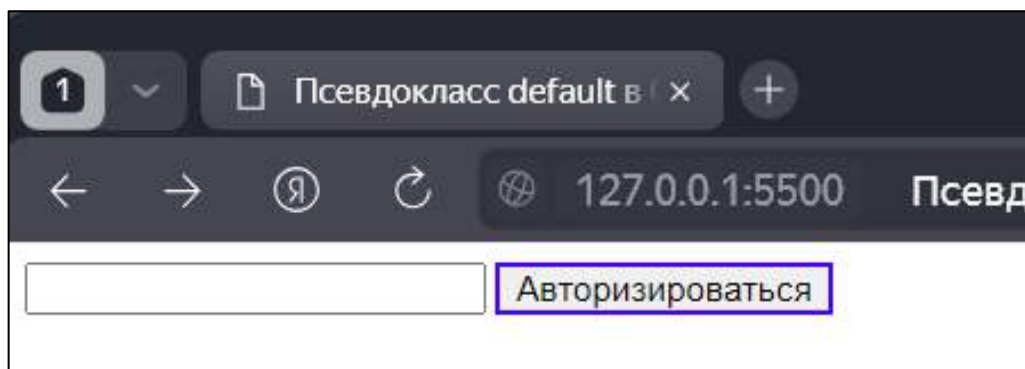


Рисунок 4.21 - Псевдокласс default в CSS 3

4.8.4 Псевдоклассы valid и invalid

Псевдоклассы :valid и :invalid стилизуют элементы формы в зависимости от того, проходят они валидацию или нет. Далее представлен пример работы с этими псевдоклассами.

```
<head> <style>
    input:invalid {border: 3px solid FireBrick;}
    input:valid {border: 3px solid ForestGreen;} </style></head>
<body><form> <h4>Почтовый адрес:</h4>
<p> Город:<input type="text" name="gorod" placeholder="Наименование города"
required /></p>
    <p>Улица:<input type="text" name="ulica" placeholder="Наименование улицы"
required /></p> <input type="submit" value="Войти" /> </form> </body>
```

Реализация данного кода представлена на рисунке 4.22.

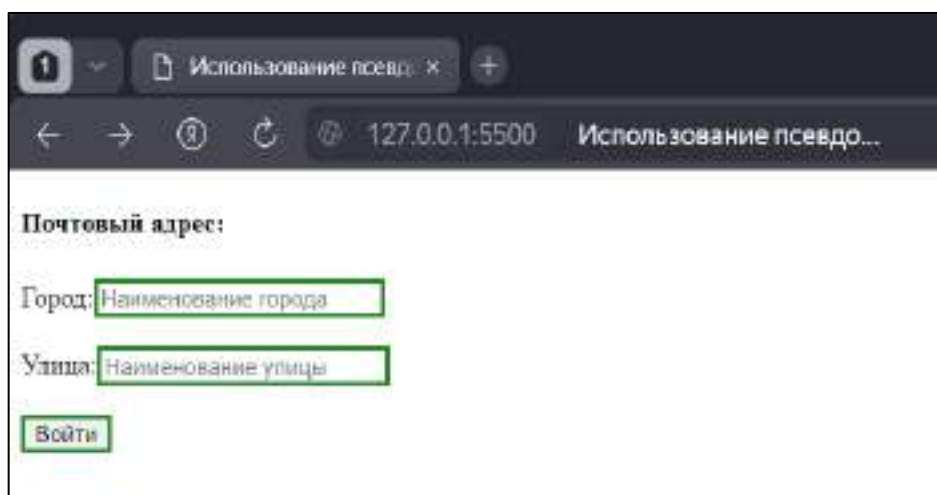


Рисунок 4.22 - Использование псевдоклассов valid и invalid в CSS3

4.8.5 Псевдоклассы in-range и out-of-range

Псевдоклассы `:in-range` и `:out-of-range` стилизуют элементы формы в зависимости от того, попадает ли их значение в определенный диапазон. Это в первую очередь относится к элементу `<input type="number" >`.

```
<head> <style>
    :in-range {border: 4px solid rgb(35, 193, 35);}
    :out-of-range { border: 4px solid rgb(247, 65, 65);} </style> </head>
<body>
<form> <p> <label for="age">Введите возраст вашей собаки:</label>
<p> <input type="number" min="1" max="20" value="1" id="age" name="age"/>
</p> <input type="submit" value="Готово" /> </form>    </body>
```

Реализация данного кода представлена на рисунках 4.23-4.24.

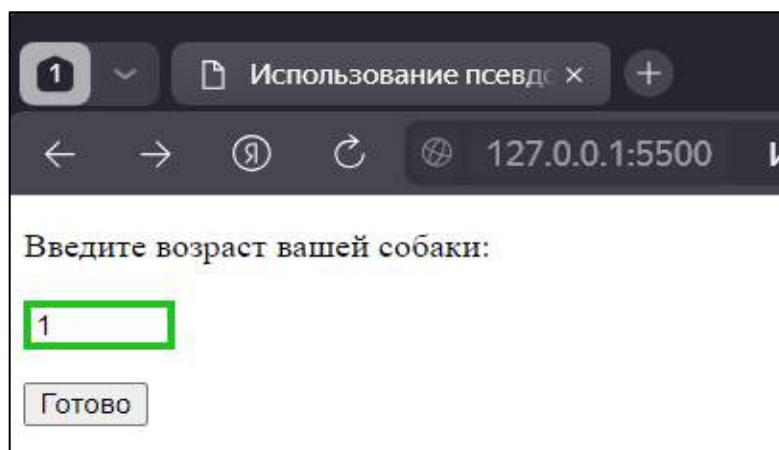


Рисунок 4.23 – Использование псевдоклассов out-of-range (корректный ввод)

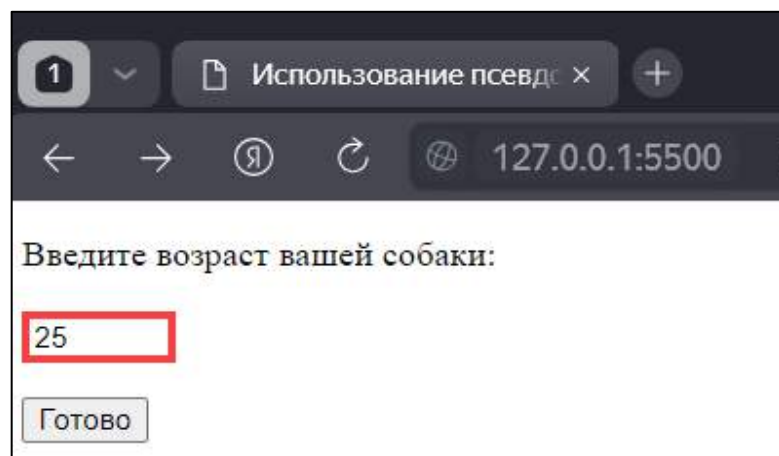


Рисунок 4.24 - Псевдоклассы out-of-range (некорректный ввод)

В данном примере атрибуты min и max задают диапазон, в которое должно попадать вводимое в поле значение:

4.9 Селекторы атрибутов

Кроме селекторов элементов также можно использовать селекторы их атрибутов. Например, есть на веб-странице несколько полей input, и необходимо окрасить в красный цвет только текстовые поля. В этом случае как раз можно проверять значение атрибута type: если оно имеет значение text, то это текстовое поле, и соответственно его надо окрасить в синий цвет. Определение стиля в этом случае выглядело бы так:

input[type="text"]{border: 3px solid blue;}

После элемента в квадратных скобках идет атрибут и его значение. То есть в данном случае для текстового поля надо установить границу синего цвета три пикселя толщиной, сплошной линией.

```
<head> <style>input[type="text"]{border: 3px solid blue;}
</style> </head>
<body> <p><input type="text" id="email" /></p>
      <p><input type="password" id="password" /></p>
      <input type="submit" value="Отправить" /> </body>
```

Реализация данного кода представлена на рисунке 4.25.

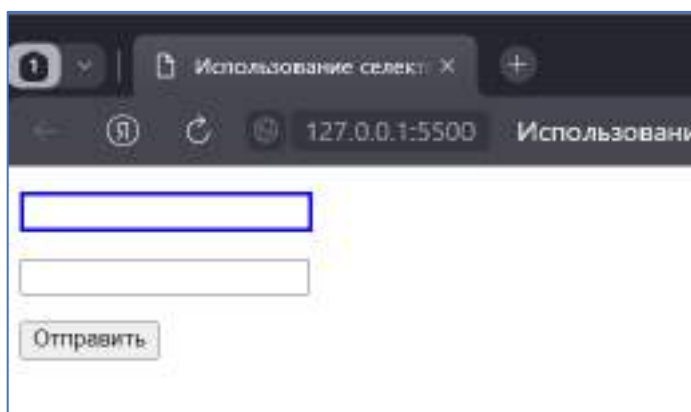


Рисунок 4.25 - Селекторы атрибутов в CSS3

Селекторы атрибутов можно применять не только к элементам, но и классам и идентификаторам. Например, в ниже представленном коде представлен пример использования селекторов атрибутов при работе с классом.

```
<head> <style> .link[href="https://ru.wikipedia.org/wiki/Музыкант"]{color: aqua;}
</style> </head>
<body> <h2>Интересное про профессии:</h2>
<a class="link" href="https://ru.wikipedia.org/wiki/Музыкант">Музыкант</a>
| <a class="link" href="https:// ru.wikipedia.org/ wiki/Пекарь"> Пекарь</a>
| <a class="link" href="https://ru.wikipedia.org/wiki/Стилист">Стилист</a>
| <a class="link" href="https://ru.wikipedia.org/wiki/Водитель">Водитель</a>
| <a class="link" href="https://ru.wikipedia.org/wiki/Психолог">Психолог</a> </body>
```

Реализация данного кода представлена на рисунке 4.26.

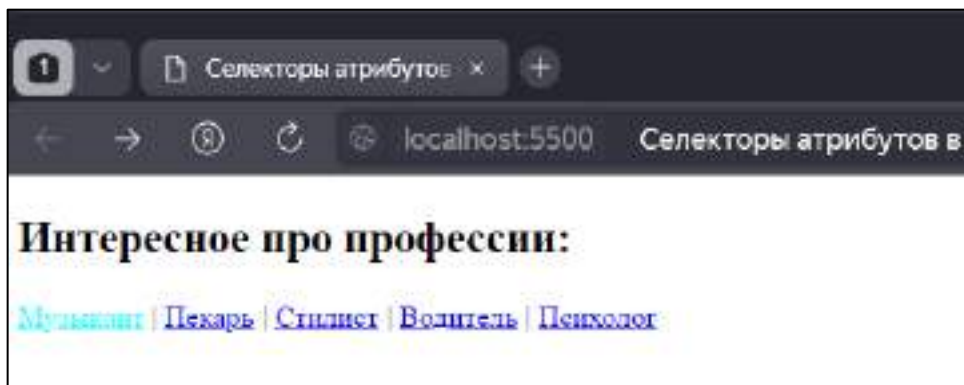


Рисунок 4.26 - Селекторы атрибутов в CSS

Специальные символы позволяют конкретизировать значение атрибутов. Например, символ `^` позволяет выбрать все атрибуты, которые начинаются на определенный текст. Если надо выбрать все ссылки, которые используют протокол `https`, то есть ссылка должна начинаться на `"https://"`. В этом случае можно применить следующий селектор:

```
a[href^="https://"]{ color: aqua; }
```

Если значение атрибута должно иметь в конце определенный текст, то для проверки используется символ `$`. Если надо выбрать все изображения в формате `jpg`, можно проверить, оканчивается ли значение атрибута `src` на текст `".jpg"`:

```
img[src$=".jpg"]{width: 100px;}
```


4.10 Наследование и каскадность стилей

4.10.1 Наследование в CSS

Для упрощения определения стилей в CSS применяется механизм наследования стилей. Этот механизм предполагает, что вложенные элементы могут наследовать стили своих элементов-контейнеров. Веб-страница имеет следующую структуру (рисунок 4.27):

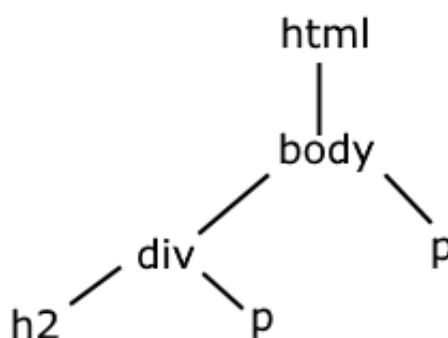


Рисунок 4.27 - Структура HTML и наследование стилей в CSS

Для элемента `div` переопределяется цвет текста. И так как элемент `h2` и один из параграфов находятся в элементе `div`, то они наследуют стиль именно элемента `div`. Второй параграф находится непосредственно в элементе `body` и поэтому наследует стиль элемента `body`.

Однако не ко всем свойствам CSS применяется наследование стилей. Например, свойства, которые представляют отступы (`margin`, `padding`) и границы (`border`) элементов, не наследуются.

4.10.2 Каскадность стилей

Когда к определенному элементу применяется один стиль, то все относительно просто. Однако если же к одному и тому же элементу применяется сразу несколько различных стилей, то возникает вопрос, какой же из этих стилей будет в реальности

применяться. В CSS действует механизм каскадности, которую можно определить как набор правил, определяющих последовательность применения множества стилей к одному и тому же элементу.

К примеру, определена следующая веб-страница, в которой применяются несколько стилей.

```
<head> <style>
    .BVLink {color: blueviolet;}
    .main a {font-weight: bold;}
    a {text-decoration: underline;}
</style> </head>
```

```
<body> <p class="main"> Каскадность - это главный принцип CSS - приоритет одних
правил/стилей над другими, когда одни стили перебивают другие. Это означает, что CSS
имеет внутреннюю иерархию, и стили с более высоким приоритетом перезаписывают правила
с более низким приоритетом.
```

```
    <a class=" BVLink" href="index.js">Подробнее узнать о каскадировании можно
узнать по ссылке</a></p> </body>
```

Реализация данного кода представлена на рисунке 4.28.

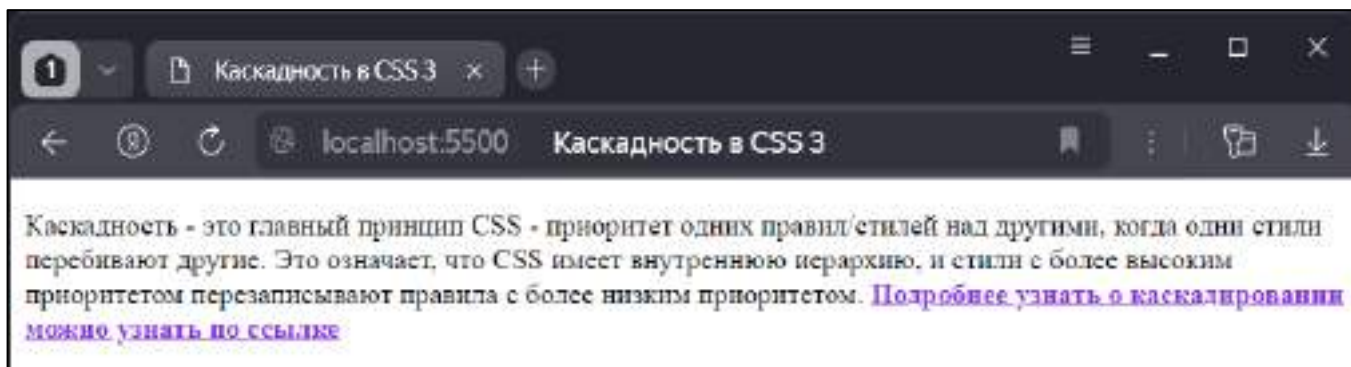


Рисунок 4.28 - Каскадность в CSS 3

В данном примере определено три стиля и все они применяются к ссылке. Если к элементу веб-страницы применяется несколько стилей, которые не конфликтуют между собой, то браузер объединяет их в один стиль. Так, в данном случае, все три стиля не конфликтуют между собой, поэтому все эти стили будут применяться к ссылке. Если же стили конфликтуют между собой, например, определяют разный цвет текста, то в этом случае применяется сложная система правил для вычисления

значимости каждого стиля. Для определения стиля к элементу могут применяться различные селекторы, и важность каждого селектора оценивается в баллах. Чем больше у селектора пунктов, тем он важнее, и тем больший приоритет его стили имеют над стилями других селекторов.

- Селекторы тегов имеют важность, оцениваемую в 1 балл.
- Селекторы классов, атрибутов и псевдоклассов оцениваются в 10 баллов.
- Селекторы идентификаторов оцениваются в 100 баллов.
- Встроенные inline-стили (задаваемые через атрибут style) оцениваются в 1000

баллов.

Далее представлен пример применения каскадирования.

```
<head> <style>
    #index {color: navy;}
    .redLink {color: ; font-size: 20px;}
    a {color: black; font-weight: bold;} </style> </head>
<body>
    <a id="index" class="redLink" href="index.js">Основы CSS 3</a>
</body>
```

В данном случае к ссылке применяется сразу три стиля. Эти стили содержат два не конфликтующих правила:

```
font-size: 20px;
font-weight: bold;
```

которые устанавливают высоту шрифта 20 пикселей и выделение ссылки жирным. Так как каждое из этих правил определено только в одном стиле, то в итоге они будут суммироваться и применяться к ссылке без проблем.

Кроме того, все три стиля содержат определение цвета текста, но каждый стиль определяет свой цвет текста. Так как селекторы идентификаторов имеют больший удельный вес, то в конечном счете будет применяться темно-синий цвет, задаваемый селектором:

```
#index {color: navy;}
```

Если селектор является составным, то происходит сложение баллов всех входящих в селектор подселекторов. Так, рассматривается следующий пример.

```

<head> <style> a {font-size: 18px;}
      .nav li a {color: red;}
      #menu a {color: navy;}
      .nav .menuItem {color: green;}
      a.menuItem:not(.newsLink) {color: orange;}
      div ul li a {color: gray;}
</style> </head><body> <h3><Меню ресторана</h3>
<div id="menu"> <ul class="nav">
<li><a class="menuItem"> Филе миньон, приготовленный на углях с картофельным
гратеном и шалотом конфи. </a></li>
<li><a class="menuItem">Говяжий язык с имбирной грушей и нежным спайси соусом.
</a></li>
<li><a class="menuItem"> Тар-Тар из говяжьей вырезки с трюфельным айоли, булочкой
бриошь и пармезаном. </a></li>
<li><a class="menuItem"> Карпаччо из говядины с трюфельной заправкой и сыром
пармезан </a></li>
<li><a class="menuItem"> Креветки васаби с авокадо, специями шичими и соусом
понзу </a></li> </ul> </div> </body>

```

Реализация данного кода представлена на рисунке 4.29.

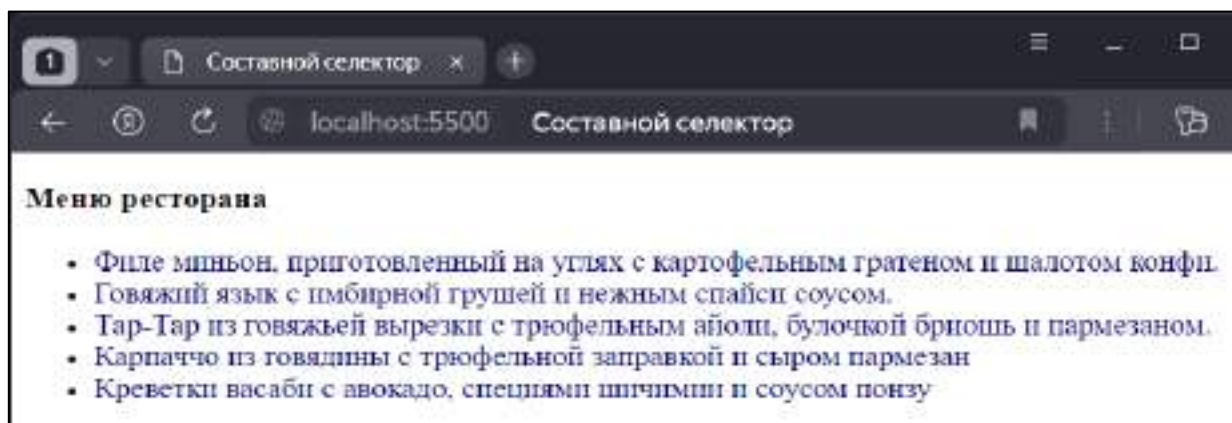


Рисунок 4.29 - Составной селектор

В данном примере определено пять различных селекторов, которые устанавливают цвет ссылок. В итоге браузер выберет селектор `#menu a` и окрасит ссылки в темно-синий цвет. Этот выбор основан на суммировании баллов по каждому из пяти селекторов, представленный в таблице 4.1.

Таблица 4.1 – Присвоение баллов селекторам

Селектор	Идентификаторы	Классы	Теги	Сумма
.nav li a	0	1	2	12
#menu a	1	0	1	101
.nav .menuItem	0	2	0	20
a.menuItem:not(.newsLink)	0	2	1	21
div ul li a	0	0	4	4

Для селектора #menu a в колонке сумма оказалось больше всего баллов - 101. То есть в нем 1 идентификатор (100 баллов) и один тег (1 балл), которые в сумме дают 101 балл. В селекторе .nav .menuItem два селектора класса, каждый из которых дает 10 баллов, то есть в сумме 20 баллов. При этом псевдокласс :not в отличие от других псевдоклассов не учитывается, однако учитывается тот селектор, который передается в псевдокласс not. CSS предоставляет возможность полностью отменить значимость стилей. Для этого в конце стиля указывается значение !important:

a {font-size: 18px; color: red !important;}

#menu a {color: navy;}

В этом случае вне зависимости от наличия других селекторов с большим количеством баллов к ссылкам будет применяться красный цвет, определяемый первым стилем.

4.11 Контрольные вопросы

- 1) Перечислить виды селекторов в CSS. Селектор класса, селектор типа, id-селектор, универсальный селектор.
- 2) Описать отличие использования внешних, внутренних и внедренных таблиц стилей.

3) Применение псевдоклассов в CSS. Привести пример псевдоклассов при работе с гиперссылками.

4) Указать отличия дочерних селекторов и селекторов потомков в CSS.

5) Рассказать о наследовании и каскадности стилей в CSS.

4.12 Тестирование по главе

1) Для определения селектора класса в CSS перед названием соответствующего класса ставится ...

- а) “-”;
- б) “.”;
- в) “.”;
- г) “#”.

2) Для определения селектора идентификатора в CSS перед названием соответствующего идентификатора ставится ...

- а) “-”;
- б) “.”;
- в) “.”;
- г) “#”.

3) Универсальный селектор представлен знаком ...

- а) “-”;
- б) “&”;
- в) “*”;
- г) “\$”.

4) Укажите верное написание при применении стиля к вложенному элементу, где main - родительский элемент (идентификатор), p – вложенный.

- а) #p main;
- б) #main p;
- в) p #main;
- г) #main *p.

5) Выберите пример селектора, который выбирает только те параграфы, которые находятся непосредственно в блоке

- а) .article p;
- б) .article < p;
- в) .article > p;
- г) .article < > p.

6) Какой псевдокласс представляет элемент, на который наведен указатель мыши?

- а) :visited;
- б) :active;
- в) :hover;
- г) :focus.

7) Какой псевдокласс выбирает элемент по умолчанию?

- а) ::enabled;
- б) ::default;
- в) ::valid;
- г) ::required.

8) Выберите правильное определения стиля в селекторе атрибутов

- а) input. [type="text"] {};
- б) input [type="text"] {};
- в) input [*type="text"] {};

г) `input." type="text"" {}`.

9) Расположите по возрастанию важность селекторов

а) идентификаторы, классы, теги;

б) теги, идентификаторы, классы;

в) теги, классы, идентификаторы;

г) идентификаторы, теги, классы.

10) На основе баллов важности селекторов, представленных в главе, посчитать баллы следующего селектора: `a #menu:not(.links)`

а) 111;

б) 101;

в) 11;

г) 222.

5 Фильтры в CSS

5.1 Назначение фильтров

Фильтры CSS создают визуальные эффекты в браузере, аналогичные фильтрам Photoshop. Для задания фильтра применяется CSS-свойство `filter`, которое принимает различные значения атрибутов, в зависимости от выбранного фильтра. Например, можно поменять цвет, яркость и насыщенность элемента, наложить размытие или изменить прозрачность, инвертировать цвета и др. Всего фильтров девять. Далее приведем примеры их использования на HTML-странице.

1) `Blur()`: значение указывается в единицах длины, например `px`, `em`. Применяет размытие по Гауссу к исходному изображению. Чем больше значение радиуса, тем сильнее размытие. Если значение радиуса не указано, по умолчанию используется значение 0.

`filter: blur(3px);`

Далее приведен пример кода с использованием фильтра `blur`.

```
<head>
  <style>
    div {width: 40%; height: 40%; margin: 20px 10px; display: flex;}
    img {width: 100%; height: 100%;}
    .filter{filter: blur(3px); margin-left: 10px;}
  </style>
</head>
<body> <div>
  
  
</div>
</body>
```

Реализация данного кода представлена на рисунке 5.1.

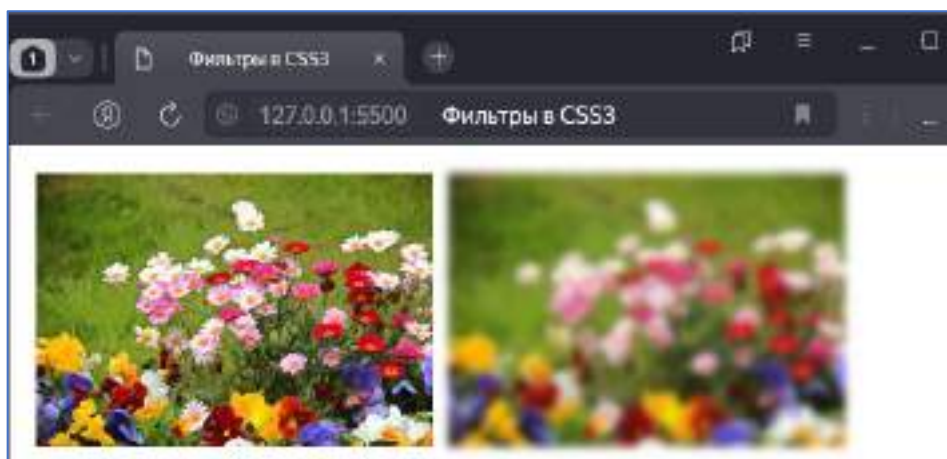


Рисунок 5.1 - Использование фильтра blur

В данном примере тег `img` имеет класс `filter`: ``, к которому применен стиль `filter: blur(3px)`.

2) `Brightness()`: значение указывается в % или в десятичных дробях. Изменяет яркость изображения. Значение по умолчанию - 1.

`filter: brightness(50%);` либо `filter: brightness(.5);`

На рисунке 5.2 представлена реализация фильтра `brightness`.

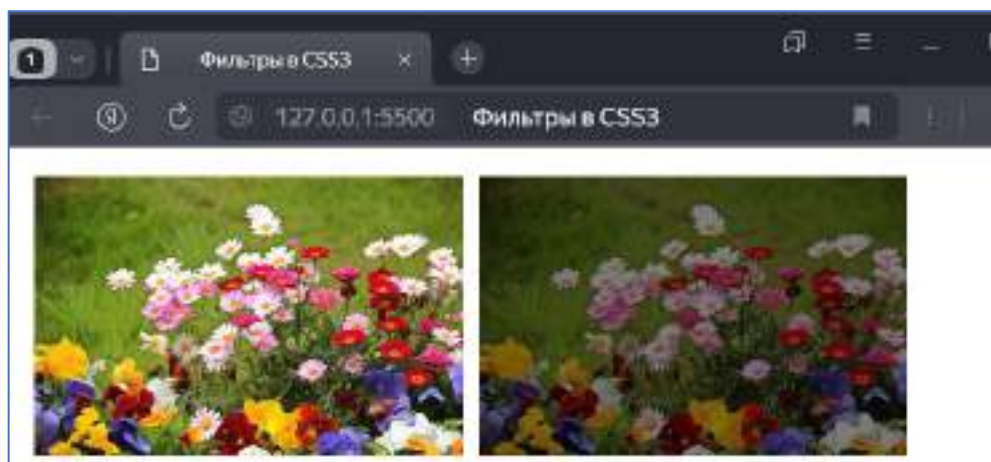


Рисунок 5.2 – Использование фильтра brightness

3) `Contrast()`: значение указывается в % или десятичных дробях. Регулирует контрастность изображения, то есть разницу между самыми темными и самыми светлыми областями изображения/фона. Значение по умолчанию - 100%. Значение 0 скроет исходное изображение под темно-серым фоном.

`filter: contrast(20%);` либо `filter: contrast(.2);`

На рисунке 5.3 представлена реализация фильтра contrast.

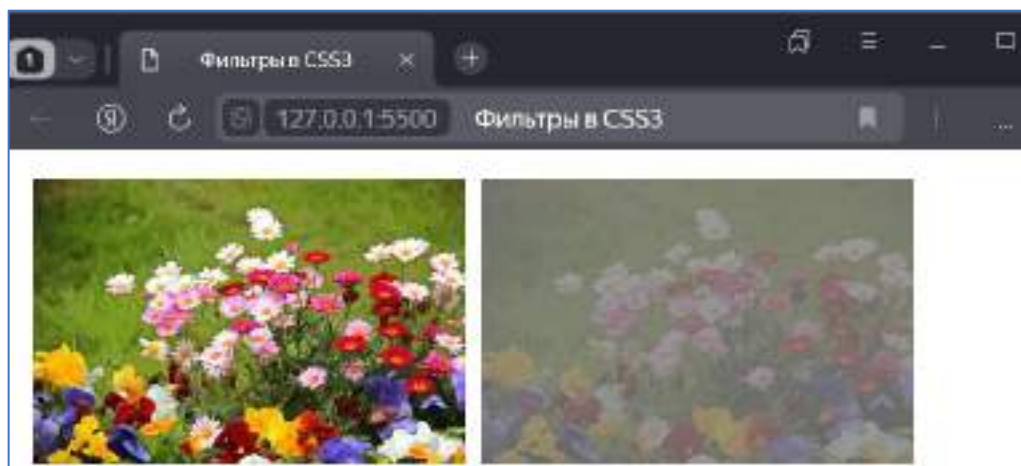


Рисунок 5.3 - Использование фильтра contrast

4) Grayscale(): извлекает все цвета из изображения, делая результат черно-белым. Значение указывается в % или десятичных дробях. Чем выше значение, тем сильнее эффект.

filter: grayscale(.5); либо **filter: grayscale(50%);**

На рисунке 5.4 представлена реализация фильтра grayscale.

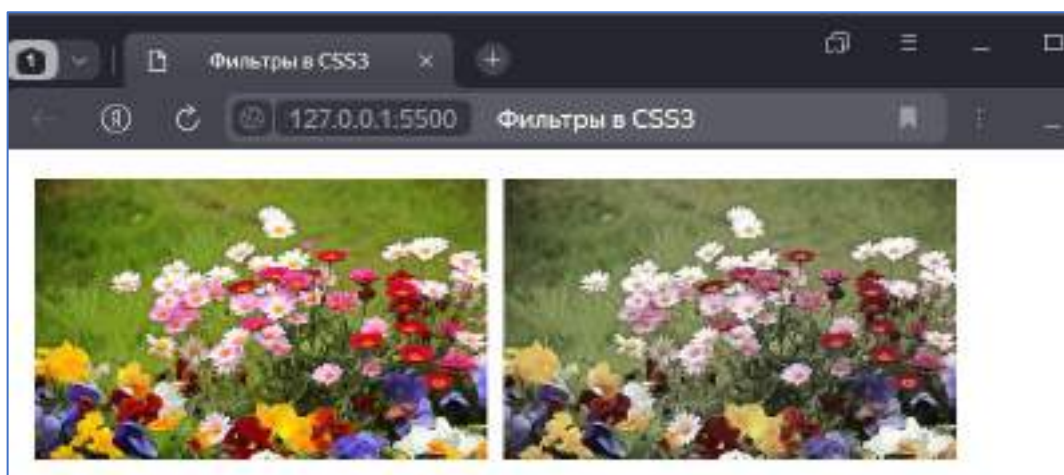


Рисунок 5.4 - Использование фильтра grayscale

5) Hue-rotate(): изменяет цвета изображения в зависимости от угла поворота, указанного в цветовом круге. Значение указывается в градусах от 0 до 360. 0deg - значение по умолчанию, означает отсутствие эффекта.

filter: hue-rotate(180deg);

На рисунке 5.5 представлена реализация фильтра hue-rotate.

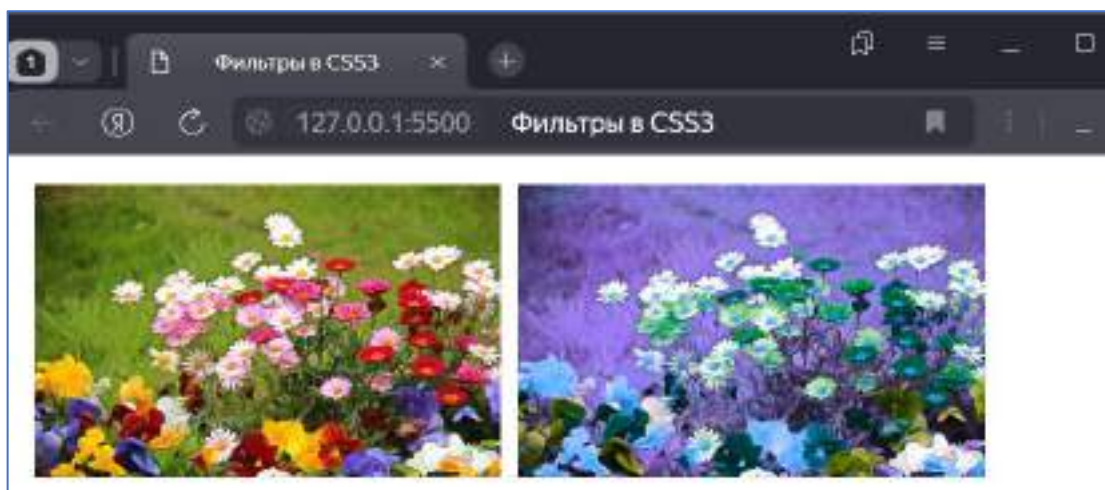


Рисунок 5.5 - Использование фильтра hue-rotate

6) `Invert()`: фильтр делает изображение негативным, инвертирует цвета. Значение указывается в %. 0% не применяет фильтр, 100% полностью преобразует цвета.

`filter: invert(100%);`

На рисунке 5.6 представлена реализация фильтра `invert`.

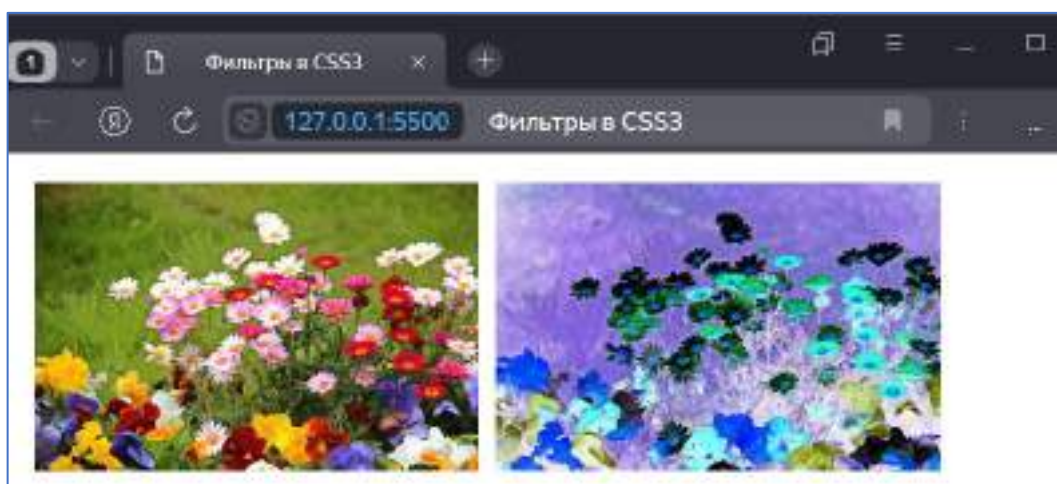


Рисунок 5.6 - Использование фильтра `invert`

7) `Opacity()`: фильтр работает аналогично свойству `opacity`, добавляя прозрачность элементу. Отличительной особенностью является то, что браузеры обеспечивают аппаратное ускорение фильтра, что повышает производительность. Дополнительным бонусом является то, что фильтр можно одновременно комбинировать с другими фильтрами, создавая интересные эффекты. Значение

задается только в %, 0% делает элемент полностью прозрачным, а 100% не имеет никакого эффекта.

filter: opacity(30%);

На рисунке 5.7 представлена реализация фильтра opacity.

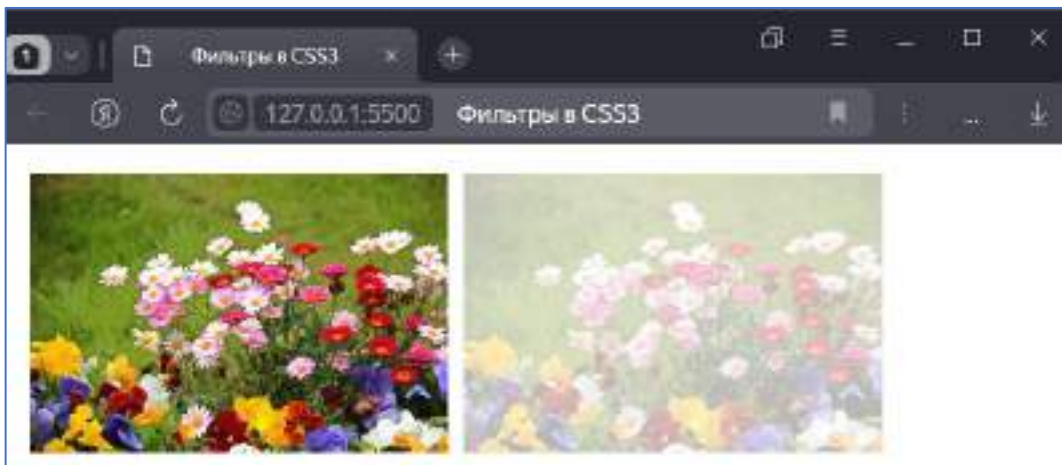


Рисунок 5.7 - Использование фильтра opacity

8) Saturate(): управляет насыщенностью цвета, действуя как фильтр контрастности. Значение 0 % удаляет цвет, а значение 100 % не оказывает никакого эффекта. Значения от 0% до 100% уменьшают насыщенность цвета, значения выше 100% увеличивают насыщенность цвета. Значение может быть указано в % или в виде целого числа, 1 соответствует 100 %.

filter: saturate(300%);

На рисунке 5.8 представлена реализация фильтра saturate.

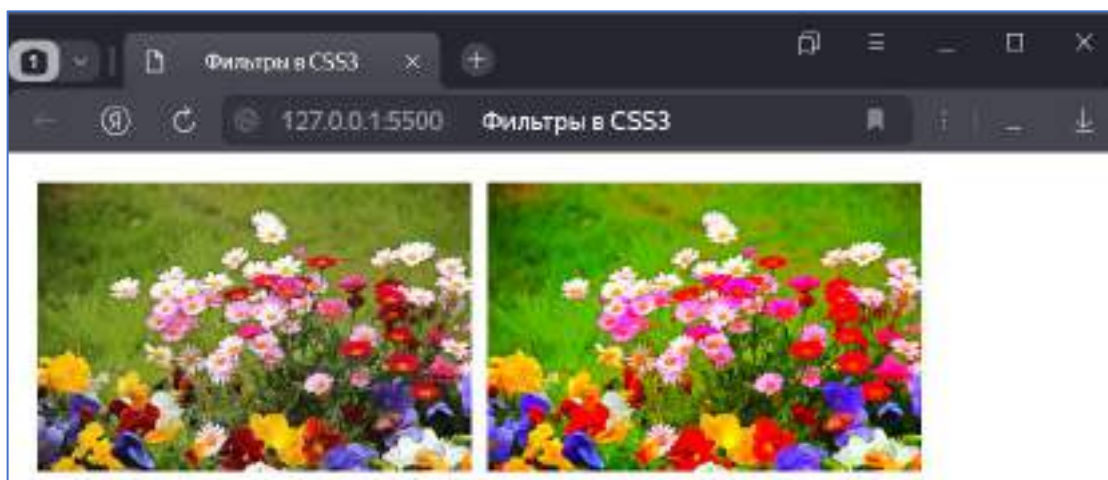


Рисунок 5.8 - Использование фильтра saturate

9) `Sepia()`: создает эффект, имитирующий старину и “ретро” фотографию. Значение 0% не меняет внешний вид элемента, а 100% полностью воспроизводит эффект сепии.

`filter: sepia(150%);`

На рисунке 5.9 представлена реализация фильтра `sepia`.

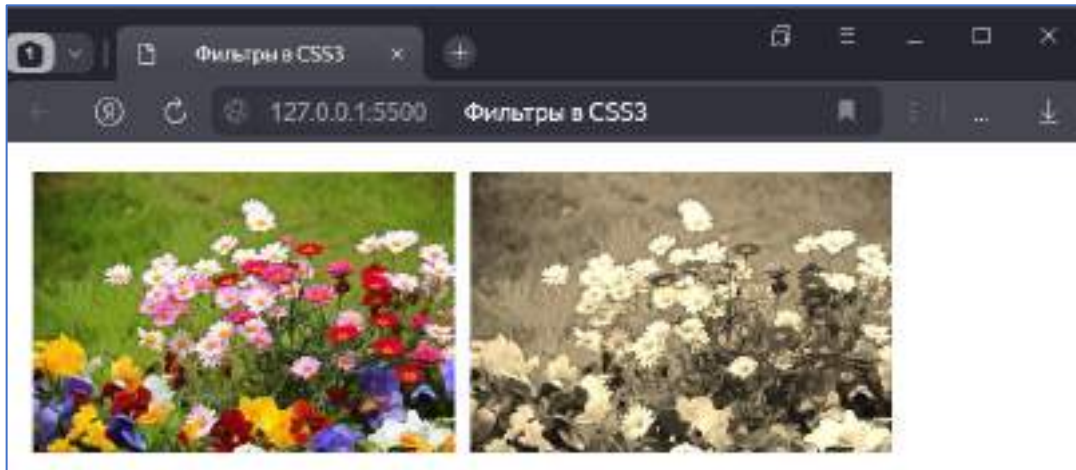


Рисунок 5.9 - Использование фильтра `sepia`

5.2 Контрольные вопросы

- 1) Рассказать об особенностях применения фильтров `blur` и `brightness`.
- 2) Рассказать об особенностях применения фильтров `contrast` и `grayscale`.
- 3) Рассказать об особенностях применения фильтров `hue-rotate` и `invert`.
- 4) Рассказать об особенностях применения фильтров `opacity` и `saturate`.
- 5) Рассказать об особенностях применения фильтра `sepia`.

5.3 Тестирование по главе

- 1) Какой фильтр в CSS делает фотографию размытой?

- a) blur,
- б) brightness;
- в) contrast;
- г) grayscale.

2) Какой фильтр в CSS изменяет яркость изображения?

- a) blur,
- б) brightness;
- в) contrast;
- г) grayscale.

3) Какой фильтр в CSS регулирует контрастность изображения?

- a) blur,
- б) brightness;
- в) contrast;
- г) grayscale.

4) Какой фильтр в CSS извлекает все цвета из изображения, делая результат черно-белым?

- a) blur,
- б) brightness;
- в) contrast;
- г) grayscale;

5) Какой фильтр в CSS изменяет цвета изображения в зависимости от угла поворота, указанного в цветовом круге?

- a) blur,
- б) brightness;
- в) contrast;
- г) hue-rotate.

6) Какой фильтр в CSS делает изображение негативным, инвертирует цвета?

а) invert;

б) opacity;

в) saturate;

г) sepia.

7) Какой фильтр в CSS работает аналогично свойству opacity, добавляя прозрачность элементу?

а) invert;

б) opacity;

в) saturate;

г) sepia.

9) Какой фильтр в CSS создает эффект, имитирующий старину и “ретро” фотографию?

а) invert;

б) opacity;

в) saturate;

г) sepia.

10) Какой фильтр в CSS аналогичен фильтру Гаусса в Photoshop?

а) blur,

б) brightness;

в) contrast;

г) grayscale.

6 Блочные элементы в CSS

6.1 Структура блочного элемента

Для веб-браузера элементы страницы представляют небольшие контейнеры или блоки, которые могут иметь различное содержимое - текст, изображения, списки, таблицы и другие элементы. Схема блочной модели представлена на рисунке 6.1.

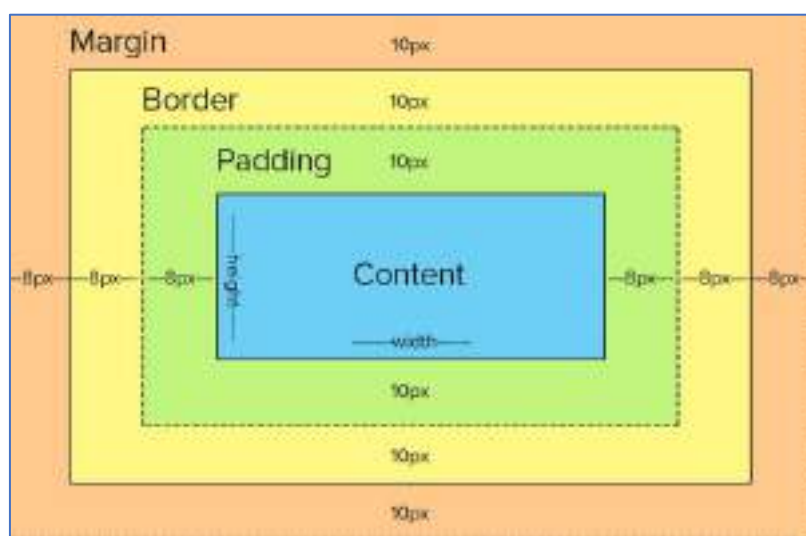


Рисунок 6.1 - Блочная модель в CSS3

Как видно на рисунке 6.1, блочный элемент имеет следующие свойства:

- `margin` - внешний отступ, определяет расстояние от границы текущего элемента до других соседних элементов или до границ внешнего контейнера,
- `padding` - внутренний отступ, определяет расстояние от границы элемента до внутреннего содержимого,
- `border` - рамка,
- `content` - внутреннее содержимое, которое также реализует ту же блочную модель и также может состоять из других элементов.

Далее рассматриваются возможности изменения свойств блочных элементов, присвоения им различных значений.

6.2 Внешние отступы

Существуют специальные свойства CSS для задания отступов для каждой стороны:

- margin-top: отступ сверху,
- margin-bottom: отступ снизу,
- margin-left: отступ слева,
- margin-right: отступ справа.

Далее представлен пример работы с блоковыми элементами, в которых заданы внешние отступы margin для разных сторон.

```
<head><style>  
div {margin-top: 30px; margin-left: 25px; border: 2px; border-color: LightSkyBlue;  
margin-right: 20px; margin-bottom: 15px; </style> </head>
```

```
<body><div> <p> Зима - самое холодное время года. В это время природа замирает до  
пробуждения весной. Повсюду лежат сугробы снега, реки озера покрываются  
льдом.</p></div>
```

```
<div> <p> Весна - самое прекрасное время года. Природа просыпается от зимней  
спячки, символизируя начало новой жизни. </p> </div> </body>
```

Реализация данного кода представлена на рисунке 6.2.

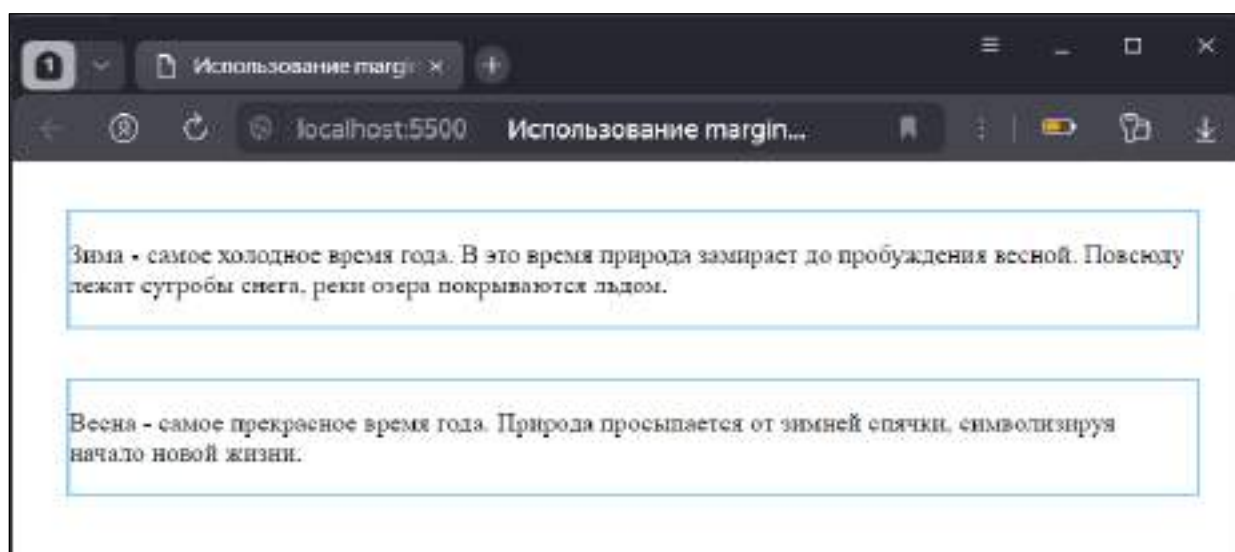


Рисунок 6.2 – Использование margin в CSS3

В данном примере вместо четырех свойств задать одно:

Div {margin: 30px 20px 15px 25px; border: 3px solid LightSkyBlue;}

Свойство задается в формате:

margin: отступ_сверху отступ_справа отступ_снизу отступ_слева;

Если значения для всех четырех отступов совпадает, то можно указать только одно значение:

div {margin: 25px;}

Для установки отступов можно использовать точные значения в пикселях (px) или em, либо процентные отношения, либо значение auto.

Например:

margin-left: 2em;

Значение 2 em определяет расстояние, которое в два раза больше размера шрифта элемента. При использовании процентов веб-браузеры вычисляют размер отступов на основе ширины элемента-контейнера, в который заключен стилизуемый элемент.

6.3 Внутренние отступы

Свойство padding задает внутренние отступы от границы элемента до его внутреннего содержимого. Как и для свойства margin, в CSS имеются четыре свойства, которые устанавливают отступы для каждой из сторон:

- padding-top: отступ сверху;
- padding-bottom: отступ снизу;
- padding-left: отступ слева;
- padding-right: отступ справа.

Далее представлен пример использования свойства padding для задания внутреннего отступа для блочного элемента.

<head> <style>

div.pad {margin: 25px; padding-top: 30px;

```
padding-right: 25px; padding-bottom: 35px;
padding-left: 25px; border: 2px solid Turquoise;}

div.in {height: 50px;}
</style> </head><body>
<div class="pad">
<div class="in"> Лето-самое жаркое из четырех времен года. В это время года дни
длиннее ночей. в этот период времени дни становятся длиннее, а ночи короче. Растения
набирают силу и радуют взгляд разнообразным буйством красок. <p> </div>
<div class="in"><p> Осень - самое грустное время года. Осень символизирует жизнь
и её цикличность, напоминая нам о быстротечности времени и о важности наслаждаться
каждым моментом. .</p>
</div> </div>
</div></body>
```

Реализация данного кода представлена на рисунке 6.3.



Рисунок 6.3 - Использование свойства padding в CSS3

Для установки значения отступов, как и в margin, могут применяться либо конкретные значения в пикселях, так и процентные значения.

Для записи отступов также можно использовать сокращенную запись:

padding: отступ_сверху отступ_справа отступ_снизу отступ_слева;

Например:

```
div.pad {margin: 25px; padding: 30px 25px 35px 28px; border: 2px solid
#0000FF;}
```

Если все четыре значения совпадают, то можно писать указать только одно значение для всех отступов:

```
div.in { margin: 25px; padding: 30px; border: 2px solid #0000FF;}
```

6.4 Границы

Граница отделяет элемент от внешнего по отношению к нему содержимого. При этом граница является частью элемента. Для настройки границы могут использоваться сразу несколько свойств: ширина, цвет и начертание.

- **border-width:** устанавливает ширину границы,
- **border-style:** задает стиль линии границы,
- **border-color:** устанавливает цвет границы.

Свойство **border-width** может принимать значения в единицах измерения, таких как **em**, **px** или **cm**, например

- **border-width: 2px;**

Также может принимать одно из константных значений: **thin** (тонкая граница - 1px), **medium** (средняя по ширине - 3px), **thick** (толстая - 5px), например

- **border-width: medium;**

Свойство **border-color** в качестве значения принимает цвет CSS:

- **border-color: red;**

Свойство **border-style** оформляет тип линии границы и может принимать одно из следующих значений:

- **none:** граница отсутствует;
- **solid:** граница в виде обычной линии;
- **dashed:** штриховая линия;
- **dotted:** линия в виде последовательности точек;
- **double:** граница в виде двух параллельных линий;
- **groove:** граница имеет трехмерный эффект;

- inset: граница как бы вдавливается во внутрь;
- outset: аналогично inset, только граница как бы выступает наружу;
- ridge: граница также реализует трехмерный эффект.

Далее представлен пример задания свойств границ для блочного элемента.

```
<head> <style> .kity {width: 100px; height: 100px; border-color: coral;
    border-width: 10px; margin-top: 10px; background-image: url(cat.png);
    background-size: cover;} </style>
</head> <body style="display: flex; justify-content: space-around">
  <div> <div class="kity" style="border-style: none">none</div>
    <div class="kity" style="border-style: solid">solid</div>
    <div class="kity" style="border-style: dashed">dashed</div>
  </div><div>
    <div class="kity" style="border-style: dotted">dotted</div>
    <div class="kity" style="border-style: double">double</div>
    <div class="kity" style="border-style: groove">groove</div>
  </div> <div>
    <div class="kity" style="border-style: inset">inset</div>
    <div class="kity" style="border-style: outset">outset</div>
    <div class="kity" style="border-style: ridge">ridge</div>
  </div> </body>
```

Реализация данного кода представлена на рисунке 6.4.

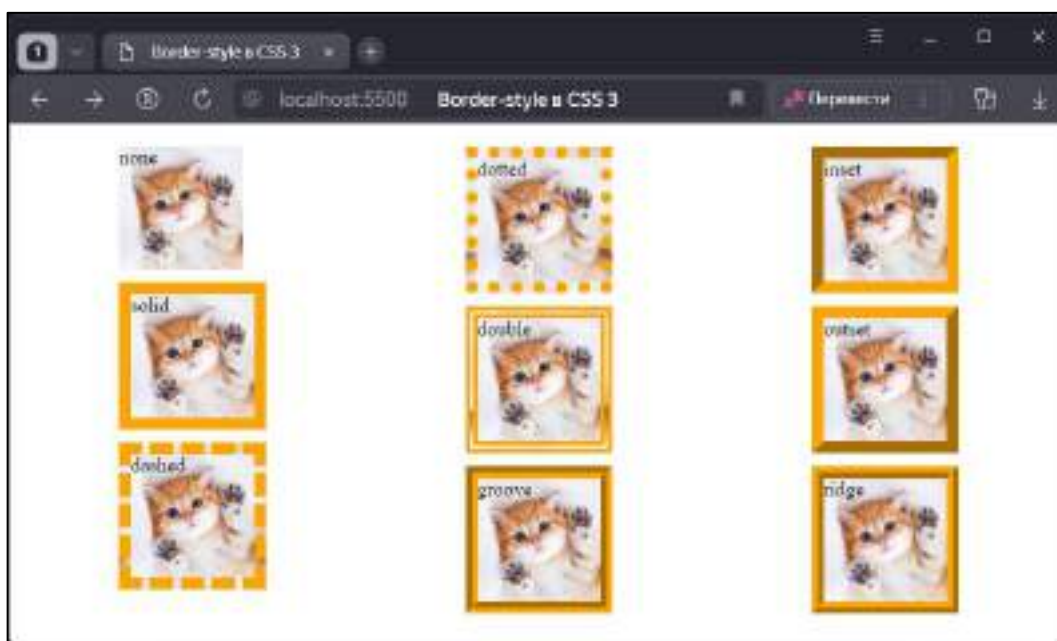


Рисунок 6.4 - Border-style в CSS 3

При необходимости можно определить цвет, стиль и ширину границы для каждой из сторон используя следующие свойства:

/ для верхней границы */*

border-top-width

border-top-style

border-top-color

/ для нижней границы */*

border-bottom-width

border-bottom-style

border-bottom-color

/ для левой границы */*

border-left-width

border-left-style

border-left-color

/ для правой границы */*

border-right-width

border-right-style

border-right-color

Вместо установки по отдельности цвета, стиля и ширины границы можно использовать одно свойство **border**.

border: ширина стиль цвет;

Например:

border: 2px solid red;

Для установки границы для отдельных сторон можно использовать одно из свойств:

– **border-top;**

– **border-bottom;**

– **border-left;**

– **border-right.**

Их использование аналогично:

border-top: 2px solid red;

border-left: 5px solid green;

При работе с границами в HTML5 есть также возможность задать ее радиус с помощью свойства `border-radius`. Другими словами, свойство `border-radius` позволяет округлить границу. Это свойство принимает значение радиуса в пикселях или единицах `em`. Далее представлен пример использования свойства `border-radius`.

```
<head>
<style>
    div {width: 100px; height:100px; border: 3px solid #8A2BE2; border-radius: 30px;}
</style> </head>
<body>
<div>Изменение радиуса границы в блоковых элементах</div>
</body>
```

Реализация данного кода представлена на рисунке 6.5.

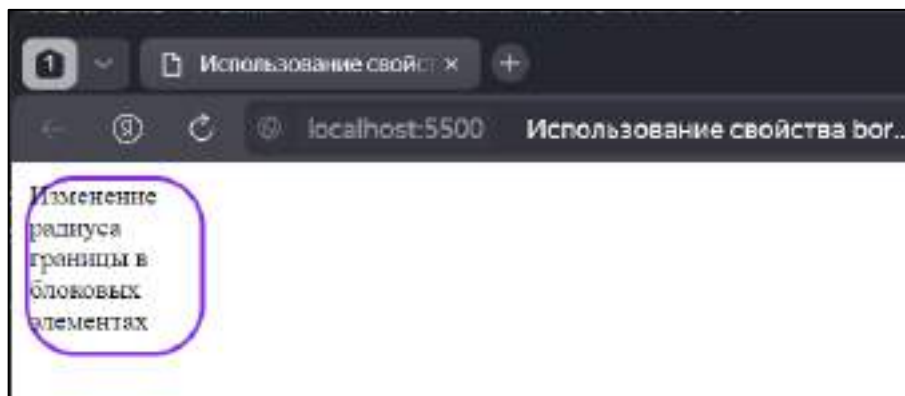


Рисунок 6.5 – Использование свойства `border-radius` для всех углов

В данном примере каждый угол будет скругляться по радиусу в 30 пикселей. Так как у элемента может быть максимально четыре угла, то можно указать четыре значения для границ **`border-radius: 15px 30px 5px 40px;`** (рисунок 6.6).

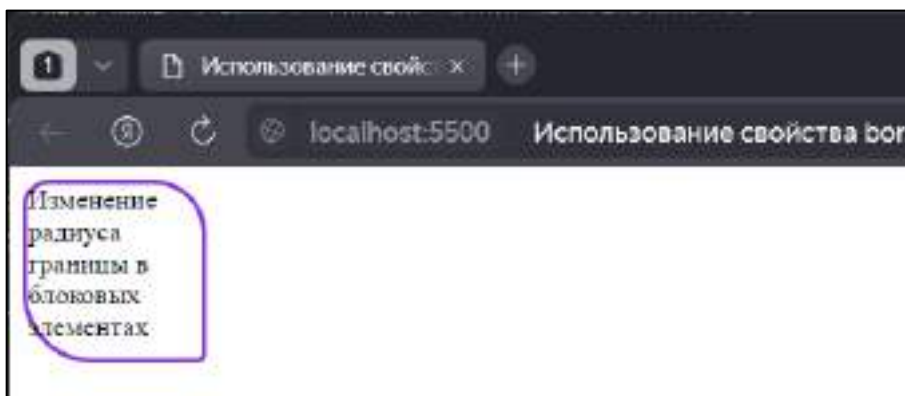


Рисунок 6.6 - Использование свойства `border-radius` для двух углов

Вместо общей установки радиусов для всех углов, можно их устанавливать по отдельности. Так, предыдущее значение border-radius можно переписать следующим образом:

border-top-left-radius: 15px; /* радиус для верхнего левого угла */

border-top-right-radius: 30px; /* радиус для верхнего правого угла */

border-bottom-right-radius: 5px; /* радиус для нижнего левого угла */

border-bottom-left-radius: 40px; /* радиус для нижнего правого угла */

Также border-radius поддерживает возможность создания эллиптических углов. То есть угол не просто скругляется, а использует два радиуса, образуя в итоге дугу эллипса (рисунок 6.7).

border-radius: 30px/20px;

В данном случае полагается, что радиус по оси X будет иметь значение 30 пикселей, а по оси Y - 20 пикселей.

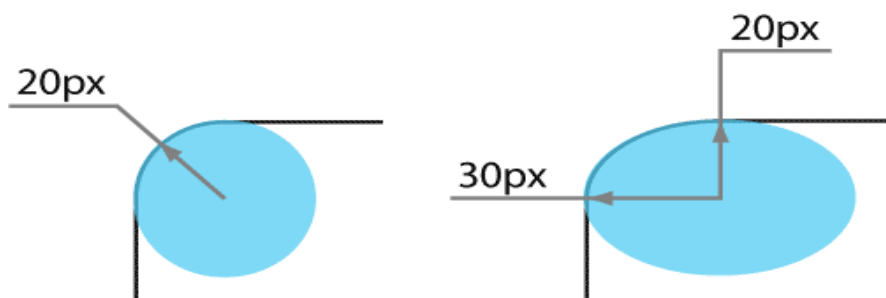


Рисунок 6.7 - Эллиптическое скругление углов в CSS 3

6.5 Фон блокового элемента

6.5.1 Установка цвета и рисунка в качестве фона элемента

Фон элемента описывается в CSS свойством background. Фактически это свойство представляет сокращение набора следующих свойств CSS:

- background-color: устанавливает цвет фона;
- background-image: в качестве фона устанавливается изображение;

- background-repeat: устанавливает режим повторения фонового изображения по всей поверхности элемента;
- background-size: устанавливает размер фонового изображения;
- background-position: указывает позицию фонового изображения;
- background-attachment: устанавливает стиль прикрепления фонового изображения к элементу;
- background-clip: определяет область, которая вырезается из изображения и используется в качестве фона;
- background-origin: устанавливает начальную позицию фонового изображения.

Далее представлен пример использования свойства background при задании фона блоковых элементов.

```
<head> <style> div {width: 250px; height: 200px; margin: 10px; }
        .colored{background-color: #00BFFF; }
        .imaged{ background-image: url(see.jpg);
        background-size: 130% }
</style> </head>
<body>
    <div class="colored">Первый блок </div>
    <div class="imaged">Второй блок</div>
</body>
```

Реализация данного кода представлена на рисунке 6.8.

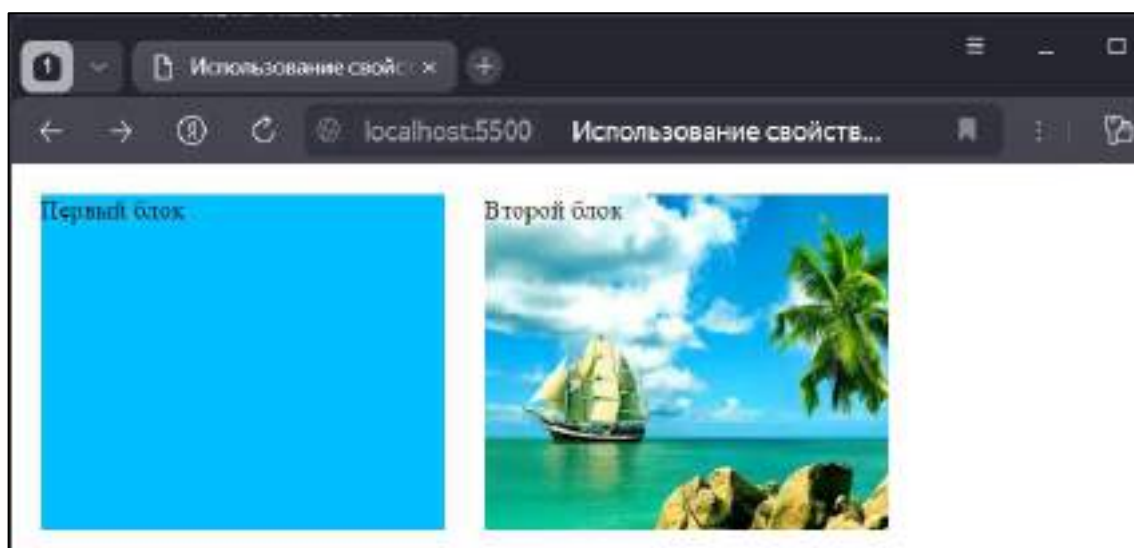


Рисунок 6.8 – Использование свойства background в CSS3

Как видно на рисунке, первый блок окрашен в оттенок синего цвета, а второй блок устанавливает в качестве фона изображение. Все содержимое блока накладывается поверх фона.

На вышеприведенном примере видно, что CSS должным образом масштабирует изображение, чтобы наиболее оптимально вписать его в пространство элемента. Однако в связи с масштабированием изображение может не полностью покрывать поверхность элемента, и поэтому для полного покрытия автоматически CSS начинает повторять изображение.

В CSS3 с помощью свойства `background-repeat` можно изменить механизм повторения. Оно может принимать следующие значения:

- `repeat-x`: повторение по горизонтали,
- `repeat-y`: повторение по вертикали,
- `repeat`: повторение по обеим сторонам (действие по умолчанию),
- `space`: изображение повторяется для заполнения всей поверхности элемента, но без создания фрагментов,
- `round`: изображение масштабируется для полного заполнения пространства,
- `no-repeat`: изображение не повторяется.

Далее представлен пример работы с фоном блокового элемента.

```
<head> <style>
    div {width: 200px; height: 150px;
    margin: 10px; border: black solid 1px;
    background-size: 140px 110px; background-image: url(dubi.png); }
    .imaged1{background-repeat: repeat-x;}
    .imaged2{background-repeat: repeat-y;}
    .imaged3{background-repeat: repeat;}
    .imaged4{background-repeat: space;}
    .imaged5{background-repeat: round;} </style> </head>
<body style="display: flex;"><div class="imaged1"></div>
    <div class="imaged2"></div> <div class="imaged3"></div>
    <div class="imaged4"></div><div class="imaged5"></div> </body>
```

Реализация данного кода представлена на рисунке 6.9.

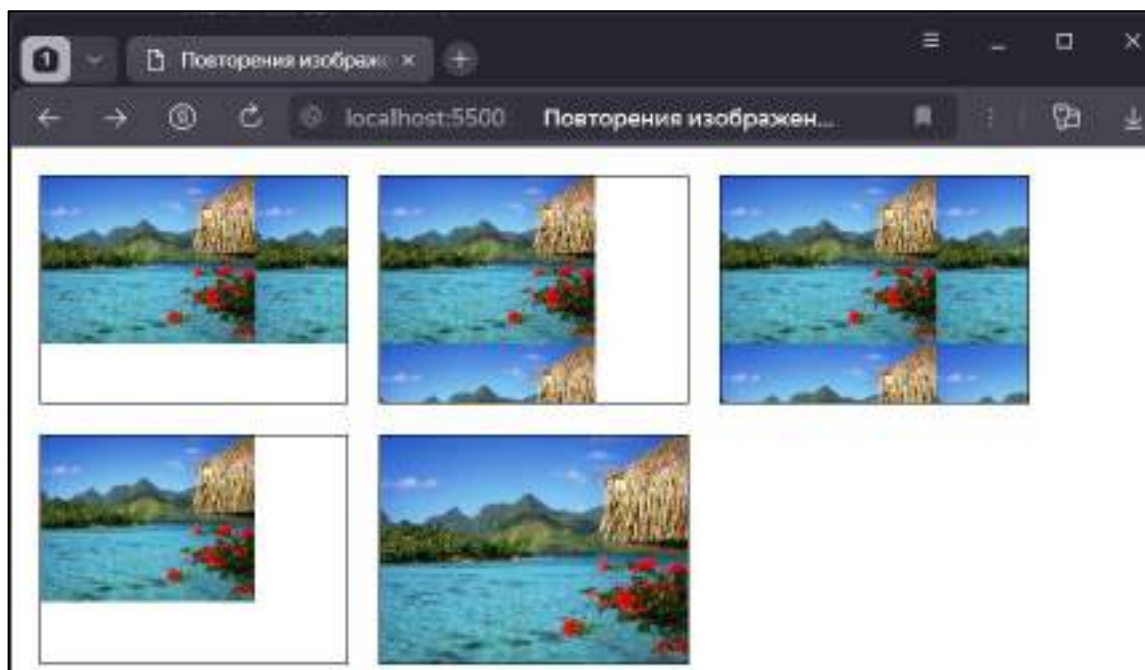


Рисунок 6.9 - Повторения изображения в блоковых элементах

6.5.2 Настройка размера фонового изображения

Помимо этого, можно изменить размер фонового изображения. Для этого следует использовать свойство `background-size`. Для установки размера можно использовать либо единицы измерения (пиксели, проценты), либо одно из предустановленных значений:

- `contain`: масштабирует изображение по наибольшей стороне,
- `cover`: масштабирует изображение по наименьшей стороне,
- `auto`: значение по умолчанию, изображение отображается в полный размер.

Если нужно масштабировать изображение, чтобы оно оптимальнее было вписано в фон, то для обеих настроек можно установить значение 100%.

`background-size: 100% 100%;`

Если задаются точные размеры, то сначала указывается ширина, а потом высота изображения.

`background-size: 200px 150px;`

Можно задать точное значение для одного измерения - ширины или высоты, для другого задать автоматические размеры, чтобы браузер выводил точные значения.

background-size: 200px auto;

Далее представлен пример настройки размеров изображений блоков.

```
<head> <style>
div {width: 200px; height: 150px; margin: 10px; border: black solid 1px;
    background-image: url(cat3.png); }
.imaged1{ background-size: cover; }
.imaged2{ background-size: 140px 110px; } </style> </head>
<body> <div class="imaged1"> </div>
<div class="imaged2"></div> </body>
```

Во втором случае изображение будет масштабироваться до размеров 140x110. Поскольку еще остается место на элементе, то по умолчанию изображение будет повторяться для заполнения всего экрана.

Реализация данного кода представлена на рисунке 6.10.

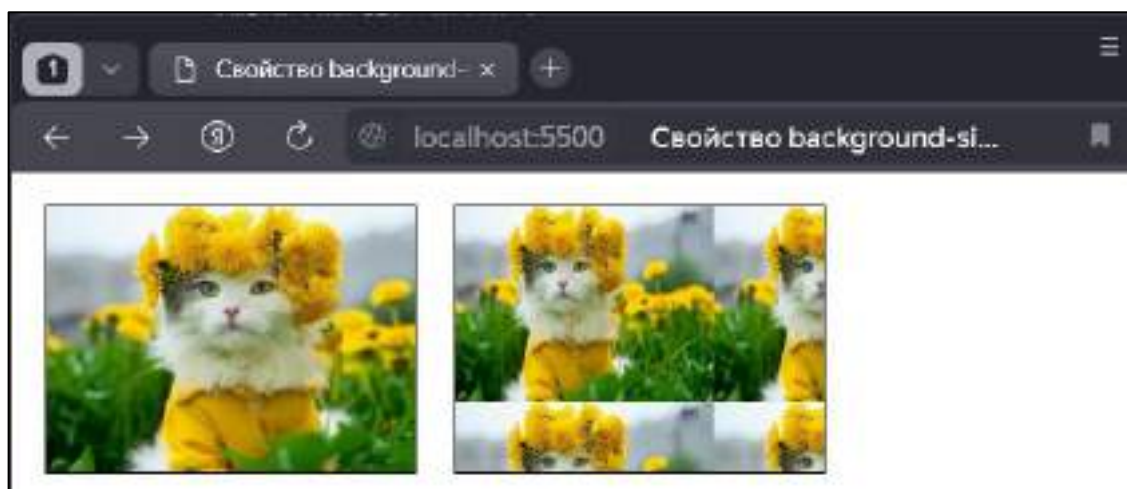


Рисунок 6.10 - Свойство background-size в CSS 3

6.5.3 Использование градиентов при оформлении фона элемента

6.5.3.1 Применение линейного градиента

Градиенты представляют собой плавный переход от одного цвета к другому. CSS3 имеет ряд встроенных градиентов, которые можно использовать для создания

фона элемента. Градиенты в CSS не представляют собой специального свойства, они просто создают значение, присваиваемое свойству background-image.

Линейный градиент проходит по прямой линии от одного конца элемента к другому, создавая плавный переход от одного цвета к другому. Для создания градиента необходимо указать его начало и несколько цветов.

background-image: linear-gradient (left, black, white);

В этом случае началом градиента является левый край элемента, заданный левым значением (left). Градиент: черно-белый. Это означает, что происходит плавный переход от черного к белому от левого края элемента к правому.

У использования градиентов есть один недостаток: разнообразие браузеров вынуждает использовать префикс провайдера:

-webkit- /* Для Yandex, Safari, Microsoft Edge, Opera выше 15 версии */

-moz- /* Для Mozilla Firefox */

-o- /* Для Opera старше 15 версии (Opera 12) */

Для установки начала градиента используются значения: left (слева направо), right (справа налево), top (сверху вниз) или bottom (снизу вверх).

Далее представлен пример использования градиентов, с использованием свойства left.

```
<head>
<style>
  div {width: 200px; height: 100px;
    background-image: linear-gradient(left, #FFFF00, #008000);
    background-image: -o-linear-gradient(left, #FFFF00, #008000);
    background-image: -moz-linear-gradient(left, #FFFF00, #008000);
    background-image: -webkit-linear-gradient(left, #FFFF00, #008000);}
  p {margin: 0; padding-top: 30px; text-align: center; color: white;} </style> </head>
<body>
<div><p>Линейный желто-зеленый градиент</p>
</div>
</body>
```

Реализация данного кода представлена на рисунке 6.11.

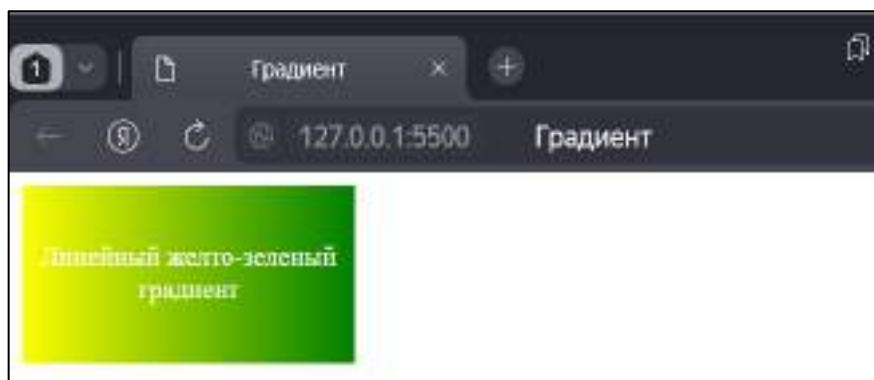


Рисунок 6.11 – Использование горизонтальных линейных градиентов

Если использовать свойство `bottom`, то градиент меняет направление на вертикальный. Например, далее представлен пример, в котором использовалось свойство `bottom` (рисунок 6.12).

`background-image: linear-gradient(bottom, #FFFF00, #008000);`

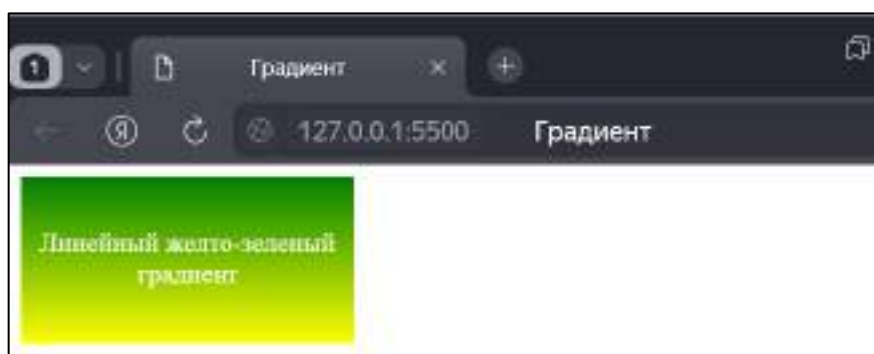


Рисунок 6.12 – Задание вертикального градиента в CSS3

Также можно задать диагональное направление с помощью двух значений (рисунок 6.13).

`background-image: linear-gradient(top left, #FFFF00, #008000);`

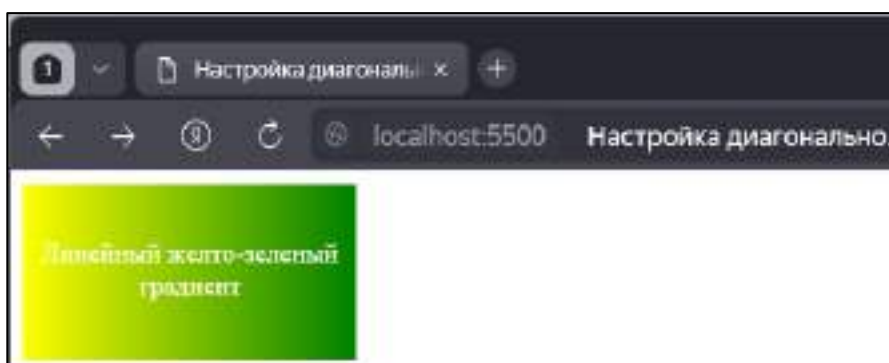


Рисунок 6.13 - Настройка диагонального линейного градиента в CSS3

Помимо конкретных значений вроде `top` или `left`, также можно указать угол от 0 до 360, который будет определять направление градиента:

```
background-image: linear-gradient(30deg, #FFFF00, blue);
```

После значения углов указывается слово `deg`. Например, `0deg` означает, что градиент начинается слева и движется вправо, а `45deg` означает, что он начинается слева внизу и движется под углом 45° вверх вправо. Определив начало градиента, можно указать цвета или опорные точки, которые нужно применить. Количество цветов может быть больше:

```
background-image: linear-gradient (top, #FFFF00, #ccc, blue);
```

Все нанесенные цвета распределяются равномерно. Можно указать определенные положения фона для цветовых точек. Для этого после цвета добавляется второе значение, определяющее положение точки (рисунок 6.14).

```
background-image: linear-gradient (left, #ccc, #FFFF00 20%, red 80%, #ccc);
```

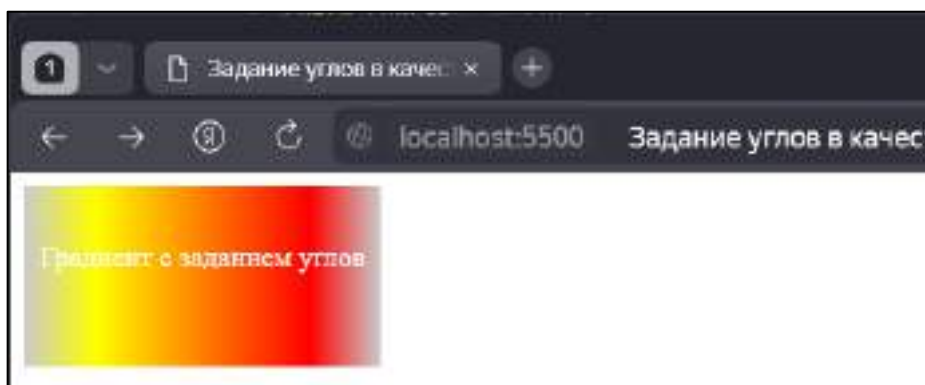


Рисунок 6.14 – Задание углов в качестве параметров градиента

6.5.3.2 Применение радиального градиента

Радиальные градиенты в отличие от линейных распространяются от центра наружу по круговой схеме. Для создания радиального градиента достаточно указать цвет, который будет в центре градиента, и цвет, который должен быть снаружи. Эти цвета передаются в функцию `radial-gradient()`.

```
<head> <style> div {width: 200px; height: 200px; border-radius: 100px;  
background-color: #eee; background-image: radial-gradient(red, blue);  
background-image: -moz-radial-gradient(red, blue);
```



```
background-image: -webkit-radial-gradient(red, blue);
p {margin: 0; padding-top: 60px; text-align: center; color: white;} </style> </head>
<body> <div><p>Радиальный градиент</p> </div></body>
```

Реализация данного кода представлена на рисунке 6.15.

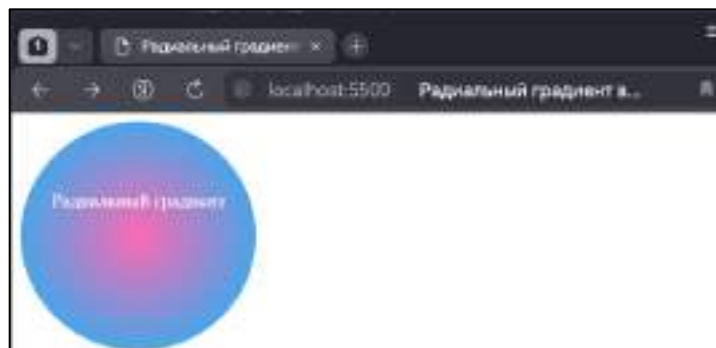


Рисунок 6.15 - Радиальный градиент в CSS3

Как и в случае с линейным градиентом здесь также надо использовать префиксы для поддержки браузерами. Радиальный градиент может иметь две формы: круговую и эллиптическую. Эллиптическая форма представляет распространение градиента в виде эллипса и задается с помощью ключевого слова `ellipse`.

```
background-image: radial-gradient(ellipse, red, blue);
```

Поскольку это значение для градиента по умолчанию, то оно может опускаться при использовании. Круговая форма представляет распространение градиента в виде кругов от центра во вне. Для этого используется ключевое слово `circle`:

```
background-image: radial-gradient(circle, red, blue);
```

Как правило, центр радиального градиента расположен в центре элемента, но это поведение можно переопределить, указав значение для параметра `background-position` (рисунок 6.16).

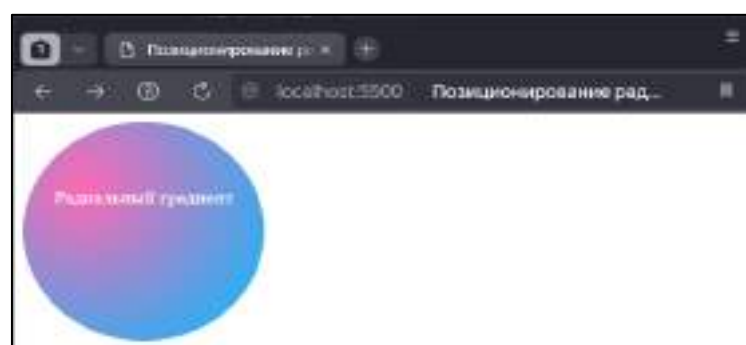


Рисунок 6.16 - Позиционирование радиального градиента в CSS3

background-image: radial-gradient(25% 30%, circle, red, blue);

Числа 25% 30% в данном случае означают, что центр градиента будет находиться на расстоянии в 25% от левой границы и в 30% от верхней границы элемента.

С помощью специальных значений можно задать размер градиента:

- closest-side: градиент распространяется из центра только до ближайшей к центру стороне элемента. То есть градиент остается внутри элемента;
- closest-corner: ширина градиента вычисляется по расстоянию из его центра до ближайшего угла элемента, поэтому градиент может выйти за пределы элемента;
- farthest-side: градиент распространяется из центра до самой дальней стороны элемента;
- farthest-corner: ширина градиента вычисляется по расстоянию из его центра до самого дальнего угла элемента.

На рисунке 6.17 представлен пример задания размера градиента.

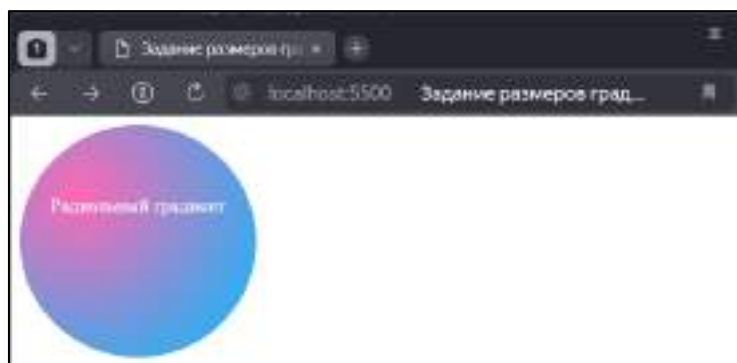


Рисунок 6.17 – Задание размеров градиенту

background-image: radial-gradient (25% 30%, circle farthest-corner, red, blue);

6.6 Позиционирование блоковых элементов

CSS предоставляет возможности размещения элементов, что означает, что есть возможность разместить элемент в определенном месте на странице.

Основным свойством, управляющим позиционированием в CSS, является свойство `position`. Это свойство может принимать одно из следующих значений:

- `absolute`: элемент позиционируется относительно границ содержащего его элемента, в качестве точки отсчета используется левое верхняя точка окна браузера,
- `relative`: элемент позиционируется относительно предыдущего элемента. В качестве точки отсчета используется левая верхняя точка окна браузера,
- `fixed`: элемент позиционируется фиксировано в окне браузера, это позволяет создавать фиксированные элементы, не меняющие положение при прокрутке,
- `static`: размещение элемента по умолчанию, значение по умолчанию.

При определении позиции элемента используются следующие параметры:

- `left` – расстояние от начала координат (от левой верхней точки окна браузера) по горизонтали, аналогично параметру `x`,
- `top` – расстояние от начала координат (от левой верхней точки окна браузера) по вертикали, аналогично параметру `y`,
- `z-индекс` – определяет порядок наложения элементов друг на друга.

6.6.1 Абсолютное позиционирование

Область отображения браузера имеет верхнее, нижнее, правое и левое поля. У каждого из этих четырех краев есть соответствующее свойство CSS: `left`, `right`, `top` и `bottom`. Значения этих свойств задаются в пикселях, `em` или процентах. Нет необходимости задавать значения для всех четырех сторон. Задаются только два значения - отступ от верхнего края -- `top` и отступ от левого края `left` (рисунок 6.18).

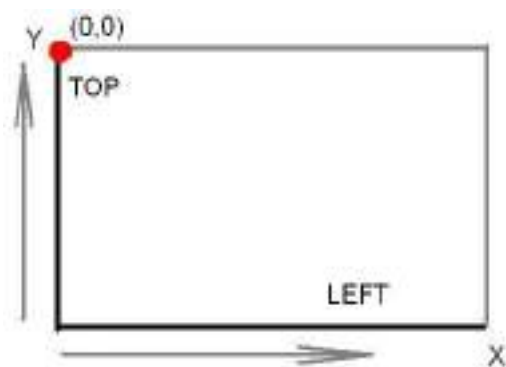


Рисунок 6.18 – Отправная точка для позиционирования элементов

Далее представлен пример применения абсолютного позиционирования для блочного элемента.

```
<head> <style> .mystyle {position: absolute;
    left: 100px; top: 50px; width: 430px; border=3;
    background-color: MediumAquaMarine;} </style> </head> <body>
```

```
<div class="mystyle"> <p>Казань – город на юго-западе России, расположенный на
берегах Волги и Казанки. В столице полуавтономной Республики Татарстан находится
древний кремль – крепость, известная своими музеями и святыми местами. Башня
Сююмбике, синие и золотые купола Благовещенского собора и яркая джума-мечеть Кул-
Шариф – одни из самых интересных достопримечательностей кремля. </p> </div> </body>
```

Реализация данного кода представлена на рисунке 6.19.

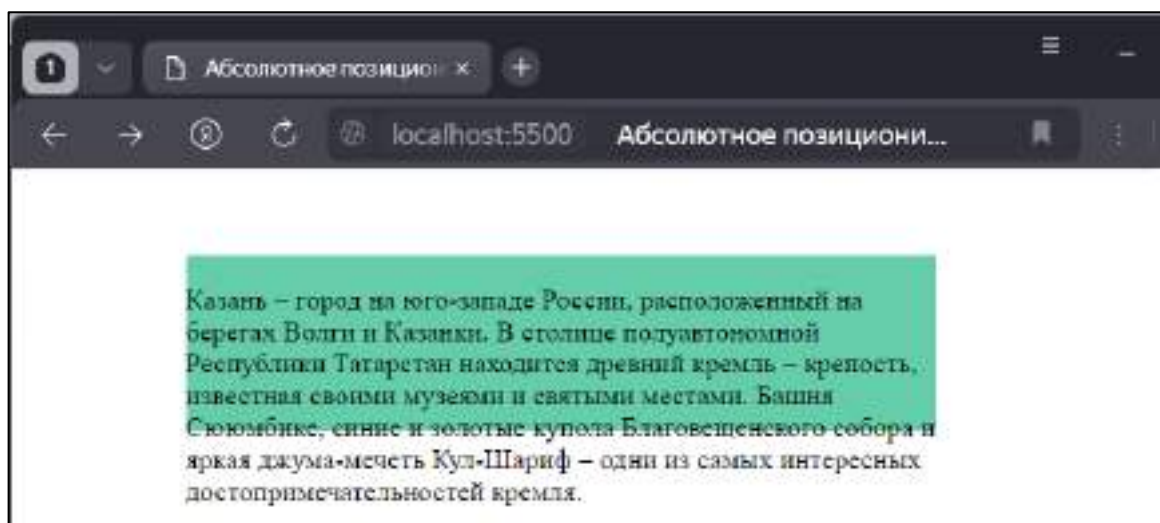


Рисунок 6.19 – Абсолютное позиционирование блочных элементов

Здесь абсолютно позиционированный элемент div будет находится на 100 пикселей слева от края окна просмотра и на 50 пикселей снизу.

Помимо форматирования блочных элементов, в CSS имеется возможность также позиционировать текст.

```
<head> <style>
    .mystyle { position: absolute; left: 100px; top: 50px; width: 430px; height: 100px;
    color: SkyBlue;} </style> </head>
```

```
<body>    <p class="mystyle"> Город-крепость Оренбург был построен в 1743 году,
расположен на слиянии двух рек - Урала и Сакмары, является главным связующим звеном
между Европой и Азией. До 1743 года Оренбург дважды закладывали в других
```

местах. Открытие газового месторождения, которое пришлось на 70-е годы XX века, явилось вторым рождением города Оренбурга. </p> </div> </body>

Реализация данного кода представлена на рисунке 6.20.

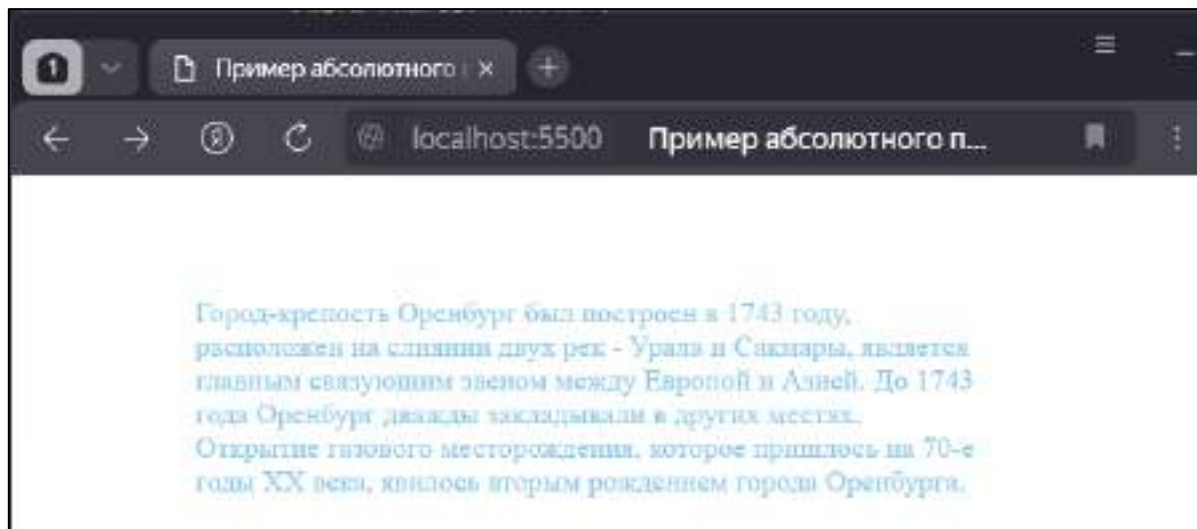


Рисунок 6.20 – Пример абсолютного позиционирования текста

6.6.2 Относительное позиционирование

Относительное позиционирование также задается с помощью свойства position со значением relative. Далее приведен пример относительного позиционирования.

```
<head> <style>
.outer {position: relative; left: 80px; top: 40px; width: 300px;
height: 100px; border: 1px solid; background-color: IndianRed ; color: white;}
.inner{ position: absolute; left: 80px; top: 40px; width: 300px;
height: 100px; border: 1px solid; background-color: SteelBlue; color: white;} </style>
</head> <body> <div class="outer">
<p> Москва - столица России и крупнейший город страны. Это огромный мегаполис,
который является историческим, политическим и духовным сердцем Российской Федерации.
Москва - крупнейшая столица Европы, наполненная достопримечательностями,
памятниками истории и культуры, а также музеями мирового уровня.</p>
<div class="inner"> <p> Санкт-Петербург - второй по численности
населения город России. Русский портовый город на побережье Балтийского моря. Был
основан в 1703 году Петром I, которому воздвигнут знаменитый памятник "Медный
всадник". Город по праву считается культурным центром страны.</p></div></div> </body>
```

Реализация данного кода представлена на рисунке 6.21.

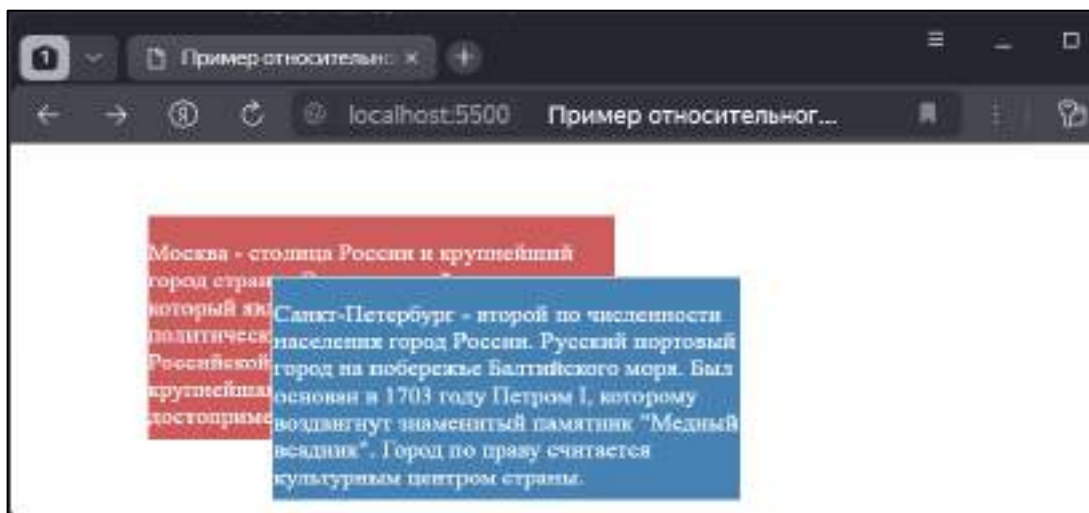


Рисунок 6.21 – Пример относительного позиционирования блоков

Аналогично, как и с абсолютным позиционированием, свойство относительного позиционирования может применяться и к текстовым элементам.

Далее представлен пример объемных заголовков с помощью абсолютного позиционирования текстовых элементов. В данном случае создается HTML-документ, в котором создаются три наложенных друг на друга надписей. Одна надпись предназначена для эффекта тени (задний план), вторая располагается над первой (передний план), еще одна используется для создания эффекта подсветки.

```
<head> <style> p {font-family:sans-serif;
    font-size:72; font-weight:800;
    color:DarkTurquoise}
p.highlight {color:silver;}
p.shadow {color:darkred;} </style></head>
<body bgcolor= LightCyan>
<div style="position:absolute; top:5; left:5">
<p class=shadow>Объемный заголовок</p></div>
<div style="position:absolute; top:0; left:0">
<p class=highlight> Объемный заголовок </p>
<div style="position:absolute; top:2; left:2">
<p>Объемный заголовок </p></div></body>
```

Реализация данного кода представлена на рисунке 6.22.

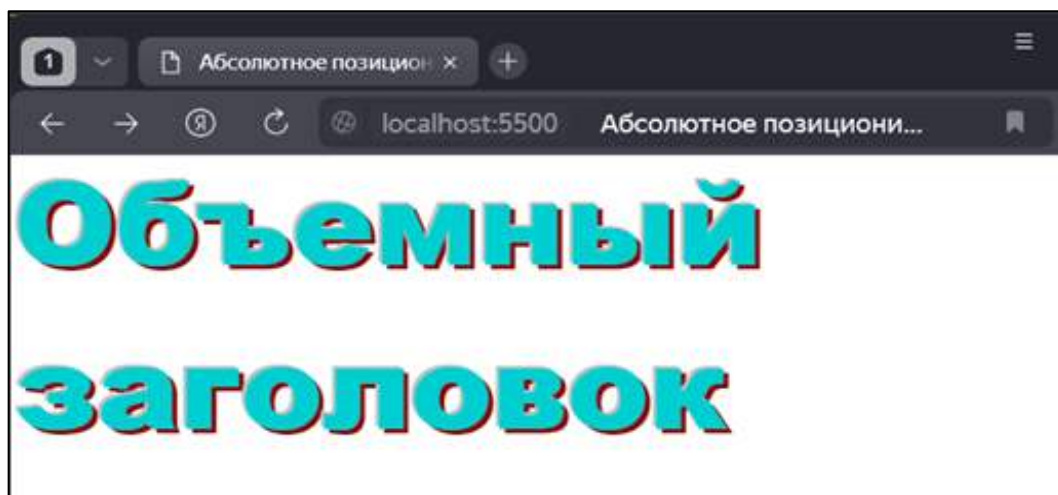


Рисунок 6.22 – Абсолютное позиционирование текста в CSS3

Таким образом, язык CSS позволяет создавать различные объемные объекты без использования графики.

6.6.3 Свойство z-index

По умолчанию, когда два элемента имеют одинаковую границу, последний элемент, определенный в разметке HTML, отображается над другим. Однако свойство z-index позволяет изменять порядок элементов при их перекрытии. Элементы с большим значением z-индекса будут отображаться над элементами с меньшим значением z-индекса. Можно поменять значения этого параметра, тем самым позволяя элементам, объявленным позднее перекрывать элементы, объявленные ранее.

Далее представлен пример работы с параметром z-index. В данном примере представлено использование перекрытия как блоковых, так и текстовых элементов.

```
<head> <style>
    .content {position: relative;
        top: 10px; left: 25px;
        width: 260px; height: 190px;
        background-color: #eee;
        border: 2px solid;}
    .LightPinkBlock {position: absolute; top: 10px; left: 40px; width: 70px;
        height: 70px; background-color: LightPink; }
```



```

.MediumVioletRedBlock { position: absolute; top: 70px; left: 70px; width: 70px; height:
70px; background-color: MediumVioletRed;}

.Block1{ position: absolute; top: 10px; left:40px; color: LightPink; }

.Block2{ position: absolute; top: 15px; left: 20px; color: MediumVioletRed; }
</style></head>

<body style="display: flex;"> <div class="content">
<div class="LightPinkBlock">
<h1> Lamborghini Aventador</h2></div>
<div class=" MediumVioletRedBlock">
<h1>Mercedes-Benz Maybach</h2></div> </div>

<div class="content" style="margin-left: 10px;">
<div class="Block1"> Lamborghini Aventador - это истинное произведение
искусства в мире автомобилестроения. Он оснащен мощным 6,5-литровым двигателем V12,
который развивает 700 лошадиных сил. Благодаря этому двигателю, Aventador способен
разогнаться до 100 километров в час всего за 2,9 секунды.</div>

<div class="Block2">Mercedes-Benz Maybach - роскошный седан, который
олицетворяет собой высшую эстетику и элегантность, возведённые в ранг абсолюта.
Полноприводная версия модели оснащена 6-ти литровым 630-сильным мотором с двумя
турбокомпрессорами. </div> </div></body>

```

Реализация данного кода представлена на рисунке 6.23.

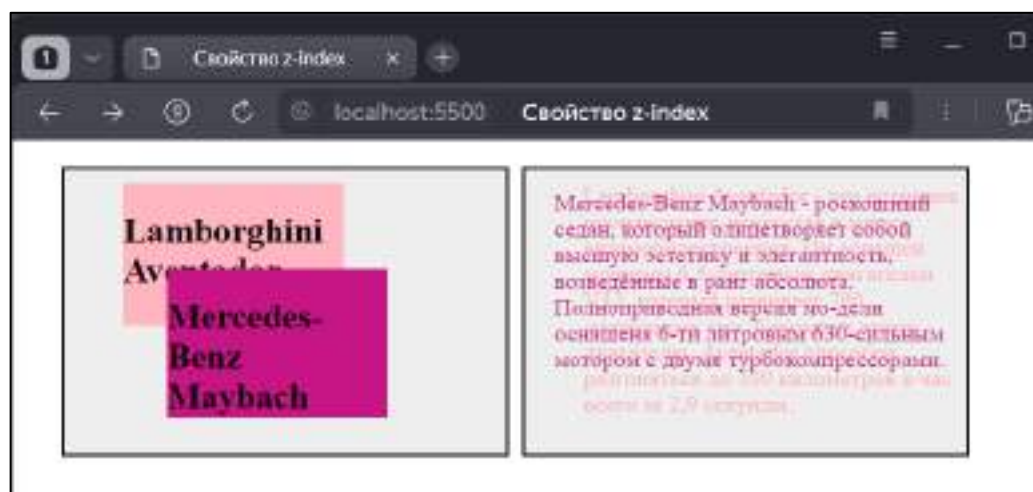


Рисунок 6.23 – Применение свойства Z-index в CSS3 для первого блока

Теперь к стилю блока добавляется новое правило:

```

.LightPinkBlock { z-index: 100; position: absolute; top: 10px; left:40px; width: 70px;
height: 70px; background-color: LightPink; }

```


Второй блок имеет неопределенный z-индекс, то для LightPinkBlock можно установить для свойства z-index любое значение, большее нуля. В данном случае 100. Теперь первый блок будет перекрывать второй, а не наоборот, как было вначале (рисунок 6.24).

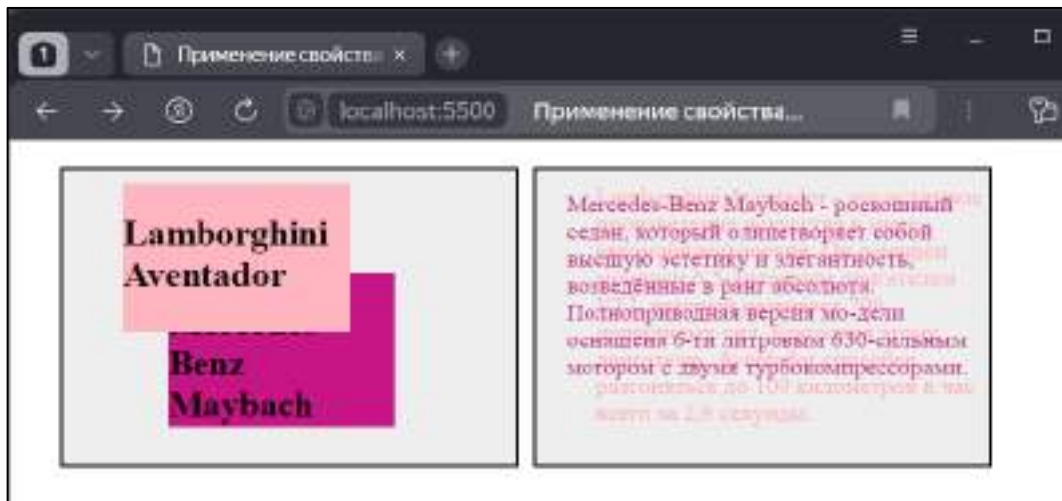


Рисунок 6.24 – Применение свойства Z-index в CSS3 для второго блока

6.6.4 Фиксированное позиционирование

Фиксированное позиционирование - это распространенный способ сохранить определенные элементы в области просмотра браузера путем создания фиксированной панели навигации, не меняющей своего положения при прокрутке.

Для фиксированного позиционирования элементов необходимо установить для свойства позиции значение “fixed”. Затем можно использовать стандартные свойства left, right, чтобы определить конкретное положение фиксированного элемента. Чтобы растянуть фиксированный блок от левой до правой границы страницы, устанавливаются три свойства: **top: 0; left: 0; right: 0;**

Далее приведен пример создания фиксированной панели навигации.

```
<head> <style>
```

```
.toolbar{ position: fixed; top: 0;left: 0; right: 0;
```

```
background-color: #4169E1;
```

```
border-bottom: 1px solid #ADD8E6;}
```

```
.toolbar a{ color: white; display: inline-block; padding: 10px; text-decoration: none;  
font-family: Verdana;}
```

```

content{ margin-top: 50px; padding: 10px;} </style> </head>
<body> <div class="toolbar">
  <a href="#">C++ </a> <a href="#">Java</a>
  <a href="#"> PHP</a> <a href="#"> Kotlin</a></div>

```

```

  <div class="content"> Сайт обучения программированию для начинающих,
который предоставляет различные образовательные ресурсы. Предоставляет множество
языков программирования для изучения и является подходящим местом как для компаний,
так и для разработчиков. </div> </body>

```

Реализация данного кода представлена на рисунке 6.25.

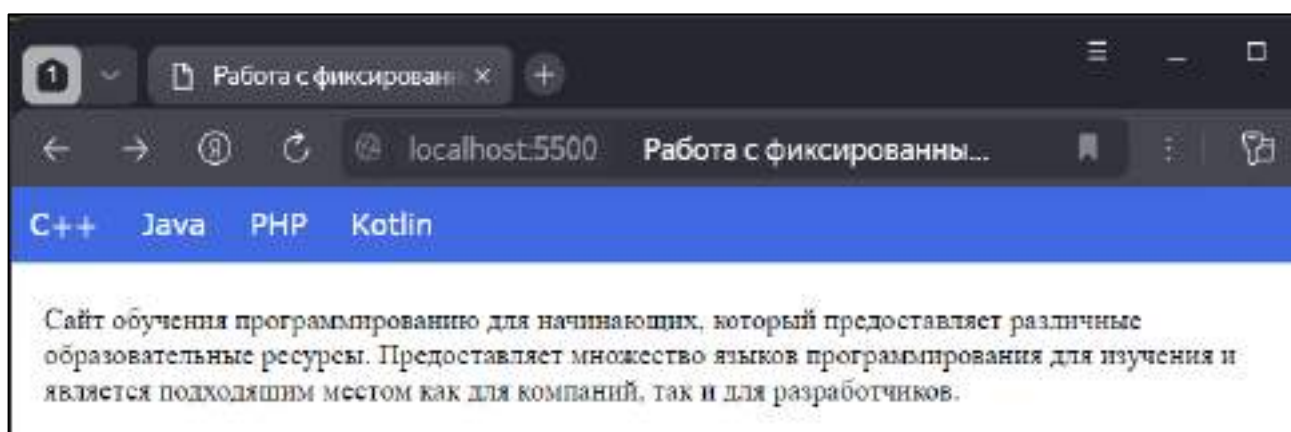


Рисунок 6.25 – Работа с фиксированными блоками в CSS

6.6.5 Статическое позиционирование

В стандартном потоке статически позиционированные элементы ведут себя аналогично относительно позиционированным: они отображаются непосредственно сразу после предыдущего элемента в потоке. Единственное их отличие от относительно позиционируемых заключается в том, что для них нельзя установить значения свойств `top` и `left`, и тем самым сместить их, например, со строки абзаца вверх или вниз.

Для того, чтобы в CSS реализовать “резиновую” верстку, были придуманы плавающие блоки. Их нельзя позиционировать с точностью до пикселя, как, например, абсолютно позиционированные. Они могут свободно перемещаться и прижиматься к краю своего контейнера. Данные блоки имеют ряд свойств, таких как `float`, `overflow`, `clear`, `display`, `visibility`. Рассмотрим каждый из них.

6.6.5.1 Свойство float

Плавающие блоки в CSS определяются свойством float. Данное свойство определяет, будет ли блок плавающим, и в какую сторону он будет перемещаться. Свойство может принимать следующие значения:

- left - структурный блок перемещается влево. Остальное содержимое документа будет выводиться вдоль правой стороны блока, начиная с самого верха.
- right - структурный блок перемещается вправо. Остальное содержимое документа выводится вдоль левой стороны блока, начиная с самого верха.
- none - блок не перемещается, т.е. позиционируется согласно алгоритму, заданному свойству position.

Далее представлен пример простого плавающего блока. Создан блок и задано свойство float со значением "right".

```
<head>
<style>
.comment { background: PaleTurquoise;
border: 1 px solid;
padding: 5 px;
width: 150 px; float: right;}
</style> </head>
<body>
```

```
<div class= "comment"> Среди самых дорогих картин Эрмитажа "Мадонна с
Младенцем" Рафаэля Санти претендует на верхнюю ступеньку пьедестала. Ранняя работа
Мастера куплена Александром II у графа Конестабиле в подарок любимой жене за 310 тыс.
франков. Собственность императрицы Марии Александровны сразу попала в выставочный
зал Эрмитажа. </div>
```

```
<p> Государственный Эрмитаж является одним из самых крупных музеев
изобразительного искусства в мире. Музей состоит из шести зданий, связанных между собой,
- Зимний дворец, Запасной дом Зимнего дворца, Малый Эрмитаж, Большой (Старый)
Эрмитаж, Новый Эрмитаж и Эрмитажный театр. Эрмитаж возник как частная коллекция
произведений искусства, приобретаемых в частном порядке российской императрицей
Екатериной II с начала её царствования. Первоначально это собрание размещалось в главной
```

императорской резиденции в специальном дворцовом флигеле - Эрмитаже (в современном комплексе - Малый Эрмитаж) - откуда и закрепилось общее название будущего музея </p></body>

Реализация данной HTML-страницы в браузере представлена на рисунке 6.26.

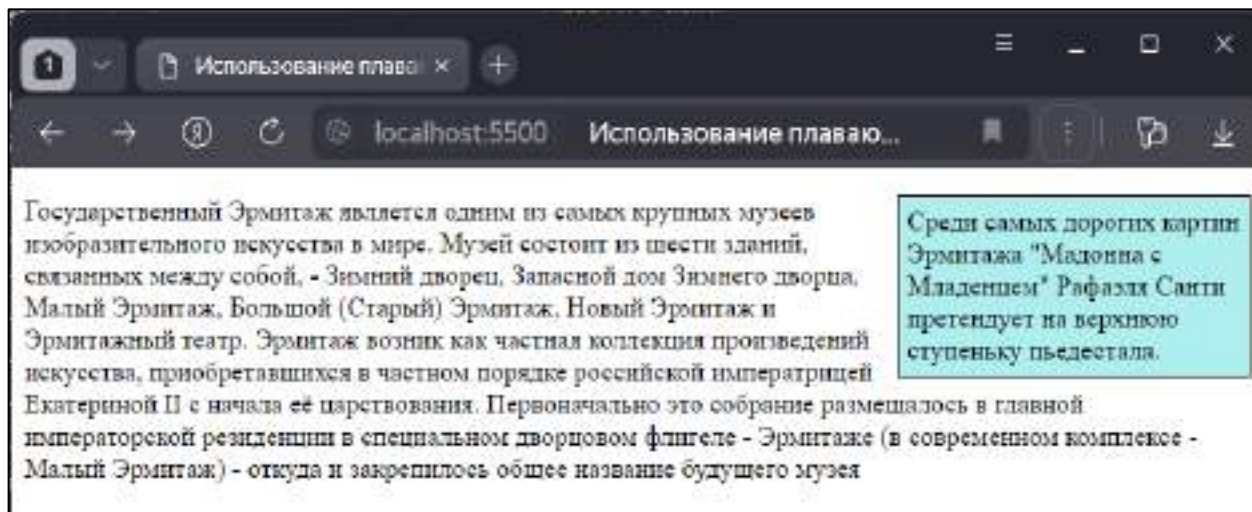


Рисунок 6.26 – Использование плавающих блоков, float: right

Как видно из рисунка 6.26, текст может обтекать не только рисунки, но и другие текстовые блоки. Если изменить значение свойства float с “right” на “left”, то блок переместится к левому краю контейнера (рисунок 6.27).

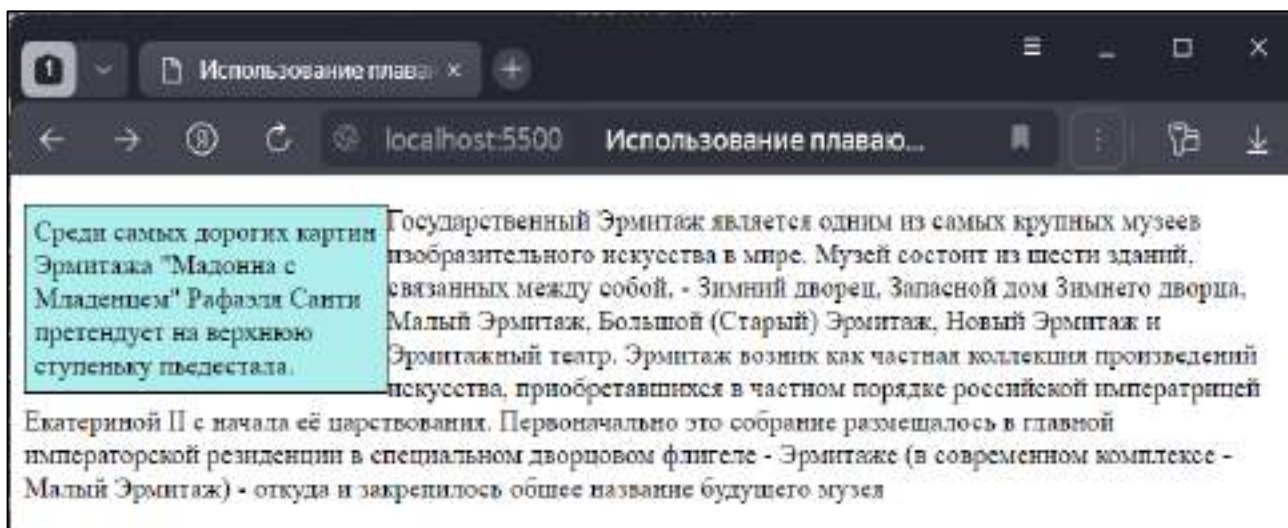


Рисунок 6.27 – Использование плавающих блоков, “float: left”

Если необходимо вставить несколько одинаковых плавающих блоков, то каждый следующий блок будет устанавливаться правее исходного (рисунок 6.28).

Как видно из рисунка второй блок располагается правее первого, несмотря на то, что также является левосторонним, но прикрепился к нему левым боком.

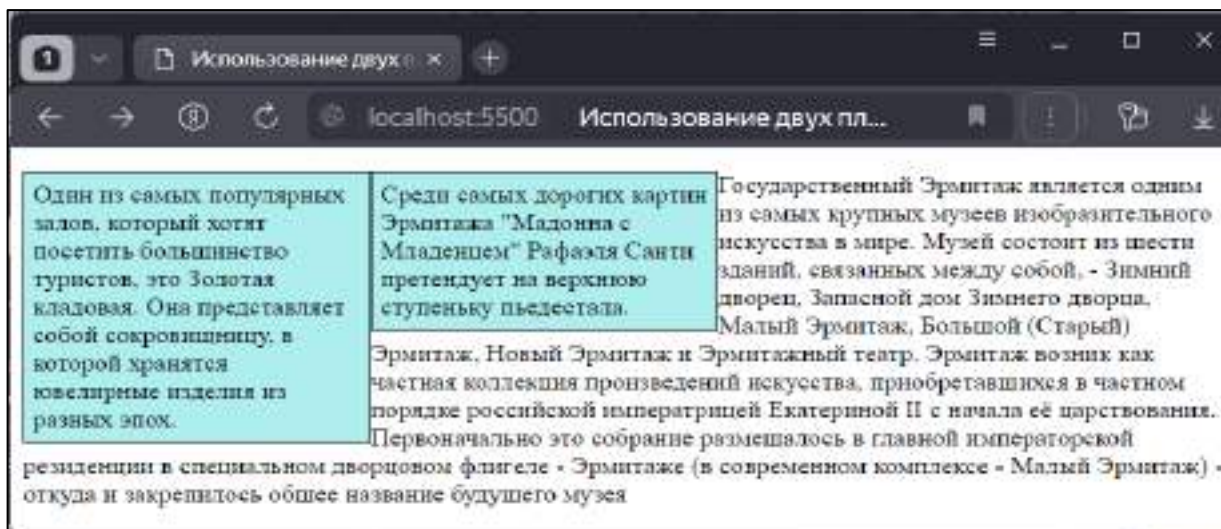


Рисунок 6.28 – Использование двух плавающих блоков

6.6.5.2 Прокрутка элементов. Свойство overflow

Нередко при создании веб-страниц можно столкнуться с ситуацией, когда содержимое блока занимает гораздо больше места, чем сам определено шириной и высотой блока. В этой ситуации по умолчанию браузер все равно отображает содержимое, даже если оно выходит за границы блока.

Однако свойство `overflow` позволяет настроить поведение блока в подобной ситуации и добавить возможность прокрутки. Это свойство может принимать следующие значения:

- `auto`: если контент выходит за границы блока, то создается прокрутка. В остальных случаях полосы прокрутки не отображаются
- `hidden`: отображается только видимая часть контента. Контент, который выходит за границы блока, не отображается, а полосы прокрутки не создаются
- `scroll`: в блоке отображаются полосы прокрутки, даже если контент весь помещается в границах блока, и таких полос прокрутки не требуется
- `visible`: значение по умолчанию, контент отображается, даже если он выходит за границы блока.

Далее представлен пример использования свойства overflow с различными значениями.

```
<head>
<style>
    .part1 {width: 300px; height: 140px; margin: 10px; border: 2px solid;
            overflow: auto; background-color:Thistle;}
    .part2 { width: 300px; height: 140px; margin: 10px; border: 2px solid;
            overflow: hidden; background-color:Thistle;}
    .part3 {width: 300px; height: 140px; margin: 10px; border: 2px solid;
            overflow: scroll; background-color:Thistle;}
    .part4 {width: 300px; height: 140px; margin: 10px; border: 2px solid;
            overflow: visible; background-color:Thistle;}
</style>
</head>
<body style="display: flex">
    <div> <div class="part1"> <p> Я не знаю, что сказать тебе при встрече. Не могу найти
    хотя бы пары слов. А недолгий вечер, а недолгий вечер. Скоро станет ночью темною без снов.
    А недолгий вечер, а недолгий вечер. Станет ночью темною без снов. </p> </div>
    <div class="part2"> <p> И снова седая ночь. И только ей доверяю я. Знает седая ночь не
    все мои тайны. Но даже и ты помочь. Не можешь, и темнота твоя. Мне одному совсем, совсем
    ни к чему. И снова седая ночь. И только ей доверяю я. Знает седая ночь не все мои тайны. Но
    даже и ты помочь. Не можешь, и темнота твоя. Мне одному совсем, совсем ни к чему. </p>
    </div> </div>
    <div> <div class="part3"> <p> Знаешь ты без слов, тебе давно все ясно. Только прячешь
    взгляд своих счастливых глаз. И опять напрасно, и опять напрасно. Собираюсь я все
    рассказать сейчас. И опять напрасно, и опять напрасно. Собираюсь все сказать сейчас.</p>
    </div> <div class="part4"> <p> И снова седая ночь. И только ей доверяю я. Знает седая
    ночь не все мои тайны. Но даже и ты помочь. Не можешь, и темнота твоя. Мне одному совсем,
    совсем ни к чему. И снова седая ночь. И только ей доверяю я. Знает седая ночь не все мои
    тайны. Но даже и ты помочь. Не можешь, и темнота твоя. Мне одному совсем, совсем ни к
    чему. </p> </div>
</div>
</body>
```

Реализация данного кода представлена на рисунке 6.29.

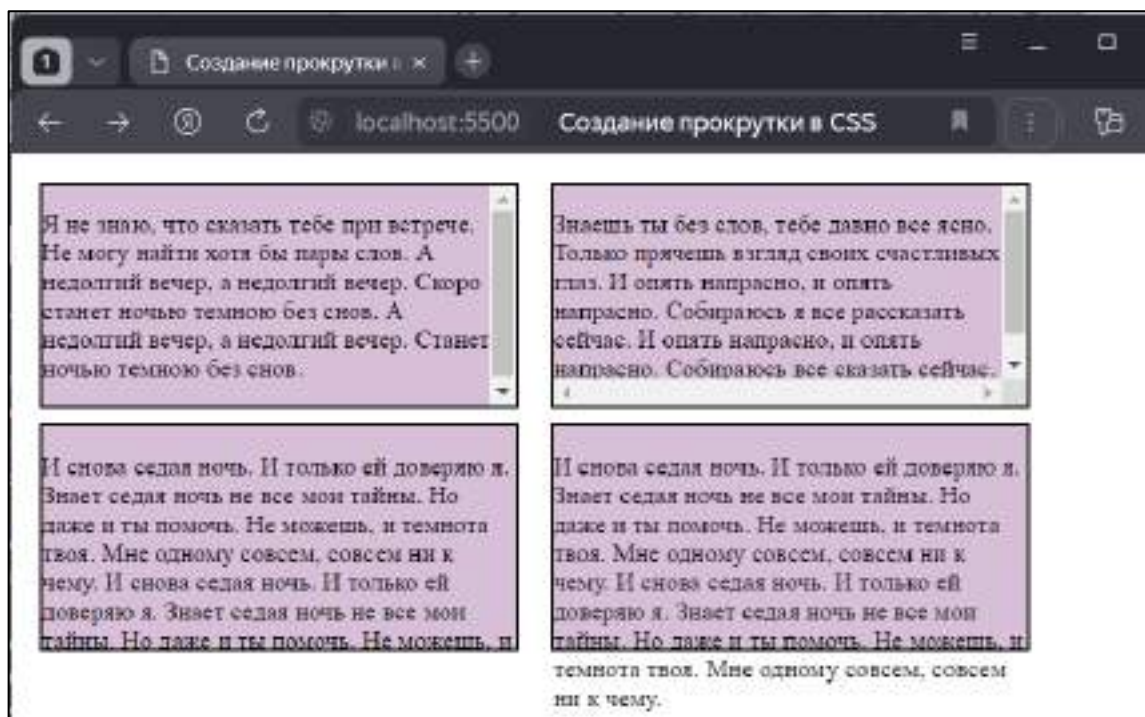


Рисунок 6.29 - Создание прокрутки в CSS

Свойство `overflow` управляет полосами прокрутки как по вертикали, так и по горизонтали. С помощью дополнительных свойств `overflow-x` и `overflow-y` можно определить прокрутку соответственно по горизонтали и по вертикали. Данные свойства принимают те же значения, что и `overflow`:

`overflow-x: auto;`

`overflow-y: hidden;`

6.6.5.3 Свойство `clear`

Свойство `clear` определяет, какие стороны перемещаемых блоков не могут соседствовать с другими перемещаемыми блоками. Свойство применяется только для элементов блочного уровня и может принимать следующие значения:

- `left` - блок должен размещаться ниже всех левосторонних плавающих блоков,
- `right` - блок должен размещаться ниже всех правосторонних плавающих блоков,
- `both` - блок должен размещаться ниже всех плавающих блоков,

- none - никаких ограничений на положение блока относительно перемещаемых объектов не накладывается.

Если у некоторого элемента существуют перемещаемые потомки, свойство clear на них не распространяется.

Далее представлен пример HTML-кода с двумя блоками, зададим для второго блока свойство “clear: left”.

```
<head>
```

```
<style>
```

```
.comment {background: SkyBlue; color: white;
```

```
border: 1 px solid; padding: 5 px;
```

```
width: 150 px; float: right;}
```

```
</style>
```

```
</head>
```

```
<body> <div class=“comment”> Одной из достопримечательностей Стамбула является  
Топкапы - главный дворец Османской империи до середины XIX века. Расположен в  
историческом центре Стамбула, на мысе Сарайбурну, в месте впадения Босфора и Золотого  
Рога в Мраморное море. </div>
```

```
<div class=“comment” style=“clear:left”> Еще одной достопримечательностью Стамбула  
является голубая мечеть. Голубая мечеть, или Мечеть Султанахмет, - первая по значению  
мечеть Стамбула. Насчитывает шесть минаретов: четыре, как обычно, по сторонам, а два  
чуть менее высоких - на внешних углах. Считается выдающимся образцом исламской и  
мировой архитектуры. </div>
```

```
<p> Самым красивым городом в Турции по праву считается Стамбул. Стамбул -  
удивительный город: огромный, шумный и очень колоритный. Он манит своей многовековой  
историей, смешением цивилизаций, сплетением европейского и азиатского. Уникальность  
географического расположения Стамбула заключается в том, что это ближайший к Европе  
азиатский город и ближайший к Азии европейский город. Стамбул одарил своей красотой не  
только Турцию, но и весь мир, став исторической сокровищницей, привлекающей внимание  
всего человечества. Стамбул – жемчужина для туристов, множество исторических мест,  
музеев, художественных галерей и экскурсий, а также хамам – все это причины, чтобы  
посетить этот чудесный город. </p>
```

```
</body>
```

Реализация данного кода представлена на рисунке 6.30.

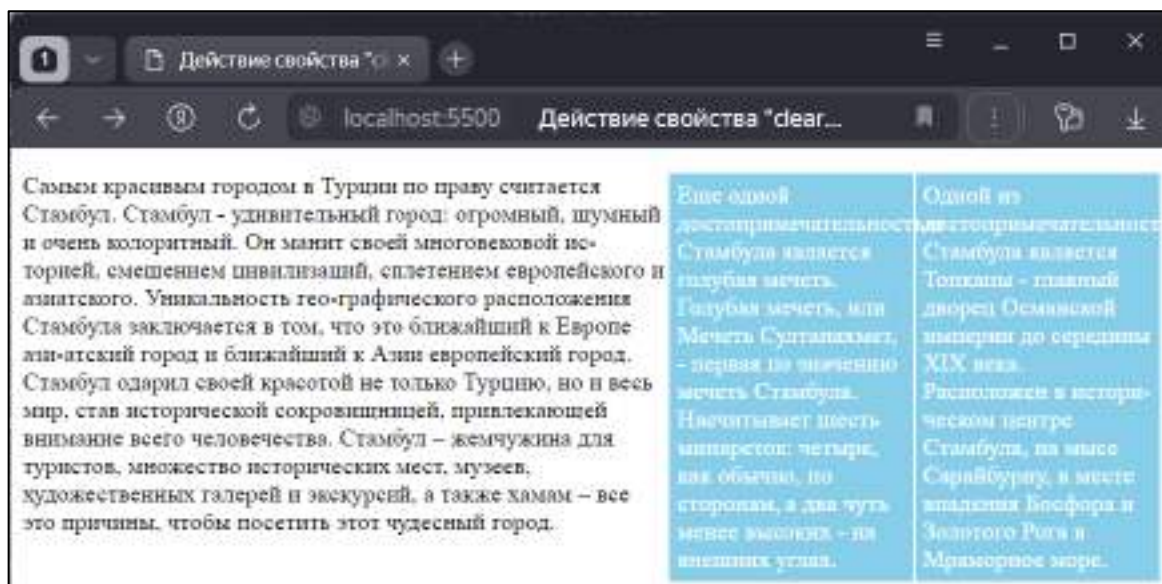


Рисунок 6.30 – Действие свойства “clear: left” на плавающий блок

Как видно из рисунка второй блок прикрепился к левому плавающему блоку снизу. Получилось два блока: один сверху, другой снизу.

Помимо этого, плавающие блоки имеют еще одну особенность. Несколько плавающих блоков будут располагаться на одной линии, если их суммарная ширина будет меньше ширины контейнера или будет равна ей. Если в качестве контейнера рассмотреть окно браузера, то таким образом можно легко реализовать верстку в несколько колонок. Ширину блоков при этом можно указать в процентах.

Далее представлен пример трех плавающих левосторонних блока так, что их суммарная ширина не превышала 100% ширины окна.

```
<head><style>
```

```
#block1 {background: PaleVioletRed; border: 1px solid;
padding: 5 px; width: 30%; float: left;}
```

```
#block2 {background: PaleVioletRed; border: 1px solid;
padding: 5 px; width: 30%; float: left;}
```

```
#block3 {background: PaleVioletRed; border: 1px solid;
padding: 5 px; width: 30%; float: left;}
```

```
</style> </head>
```

```
<body>
```

```
<div id=“block1”> Dart - это язык программирования, используемый для создания
кроссплатформенных приложений с помощью фреймворка Flutter. </div>
```

`<div id="block2">Java - один из популярных языков программирования для Android. Это объектно-ориентированный язык, который используется для создания приложений с высокой производительностью.</div>`

`<div id="block3"> Kotlin – высокоуровневый язык программирования общего применения, который является надстройкой над Java. Язык используется в качестве основного языка для разработки приложений на Android в официальной SDK – Android Studio.</div></body>`

Реализация данного HTML-кода представлена на рисунке 6.31.

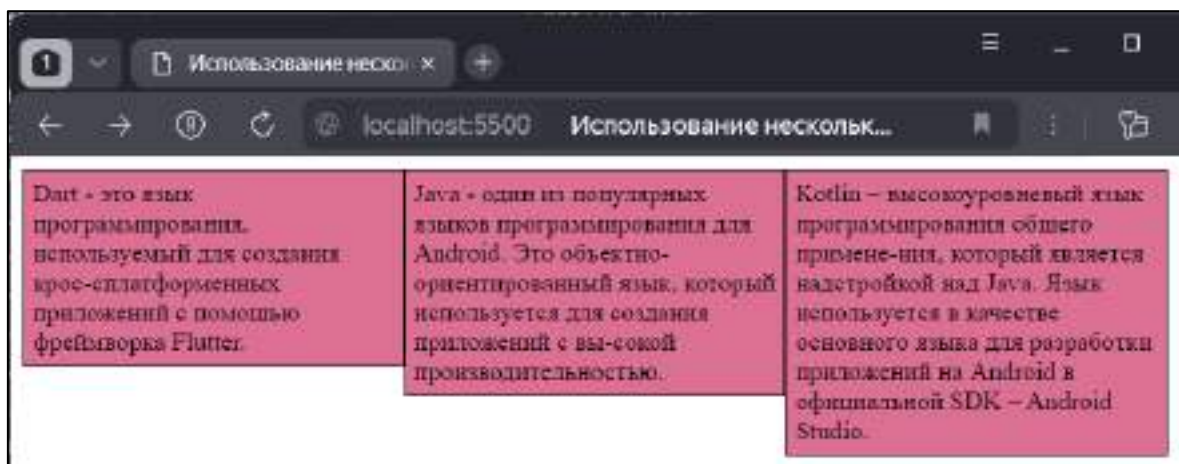


Рисунок 6.31 – Использование нескольких блоков, высота каждого зависит от его содержимого

Как видно из рисунка 6.31, высота колонок будет зависеть от содержимого блоков. Чтобы подравнивать колонки по высоте, следует в CSS-правилах задать свойство `height:100%`. Это позволяет реализовать “резиновую” (рисунок 6.32).

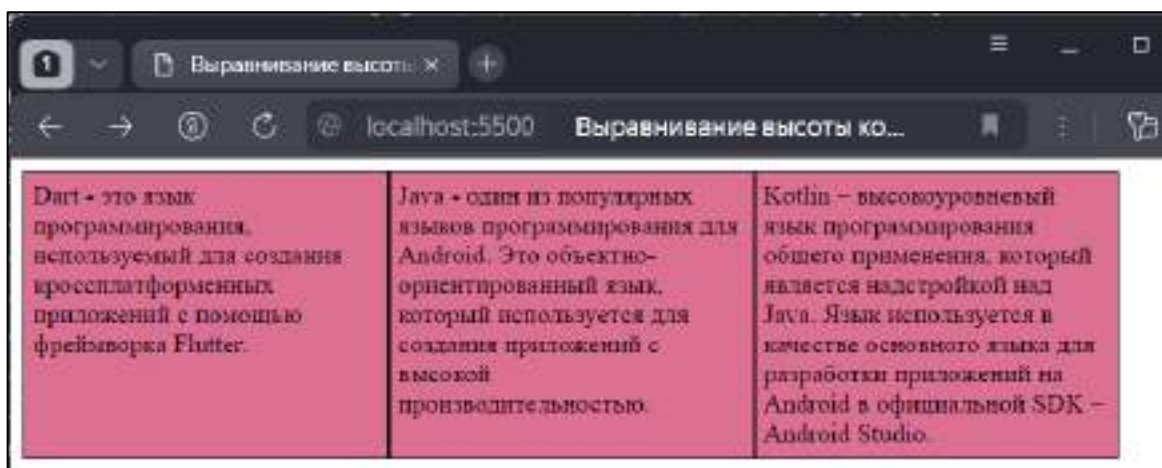


Рисунок 6.32 – Выравнивание высоты колонок с помощью свойства `height:100%`

Как видно на рисунке 6.32 высота колонок выровнена.

6.6.5.4 Свойство display

Кроме свойства float, которое позволяет изменять позицию элемента, в CSS есть еще одно важное свойство - display. Оно позволяет управлять блоком элемента и также влиять на его позиционирование относительно соседних элементов.

Это свойство может принимать следующие значения:

- inline: элемент становится строчным, подобно словам в строке текста;
- block: элемент становится блочным, как параграф;
- inline-block: элемент располагается как строка текста;
- list-item: элемент позиционируется как элемент списка обычно с добавлением маркера в виде точки или порядкового номера;
- run-in: тип блока элемента зависит от окружающих элементов;
- flex: позволяет осуществлять гибкое позиционирование элементов;
- table, inline-table: позволяет расположить элементы в виде таблицы;
- none: элемент не виден и удален из разметки html.

Итак, значение block позволяет определить блочный элемент. Такой элемент визуально отделяется от соседних элементов переносом строки, как, например, элемент параграфа p или элемент div, которые по умолчанию являются блочными и при визуализации веб-страницы визуально переносятся на новую строку. Однако элемент span в отличие от элемента div блочным не является. Далее приведены примеры работы свойства display с различными значениями.

1) Использование свойства display со значением block.

```
<head> <style>      span { color: RoyalBlue;}
      .blockSpan { display: block; border: 2px;}
</style> </head>
<body>
  <div>Черное  <span>море</span> - внутреннее море бассейна Атлантического
океана.  Проливом Босфор соединяется с Мраморным морем, далее, через
пролив Дарданеллы с Эгейским и Средиземным морями. </div>
```

```

<div>Средиземное<span class="blockSpan">море</span> - межматериковое
море Атлантического океана, между Европой, Африкой и Азией. Оно названо так
благодаря своему положению среди земель, составлявших весь известный древним мир.</div>
</body>

```

Реализация данного кода представлена на рисунке 6.32.

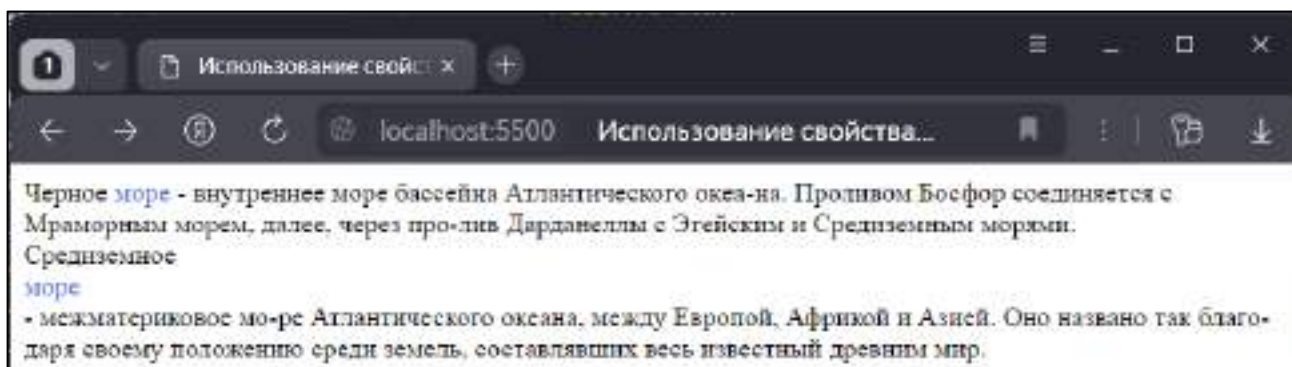


Рисунок 6.32 - Использование свойства display со значением block

В данном примере определено два элемента span, но один из них является блочным, так как к нему применяется стиль display: block;. Поэтому этот элемент span переносится на новую строку.

2) Использование свойства display со значением inline. В отличие от блочных элементов строчные встраиваются в строку, так как имеют для свойства display значение inline. Элемент span как раз по умолчанию имеет стиль display: inline, поэтому и встраивается в строку, а не переносится на следующую, как параграфы или div. Далее реализована обратная процедура - сделать блочный элемент div строчным.

```

<head>
<style>
  div{ display: inline; }
</style> </head>
<body color: FireBrick; >
<h3>Интернет магазин быстрой еды</h3>
  <div>Гамбургеры</div>
  <div>Картошка фри</div>
  <div>Нагетсы</div>
  <div>Напитки</div></body>

```

Реализация данного кода представлена на рисунке 6.33.

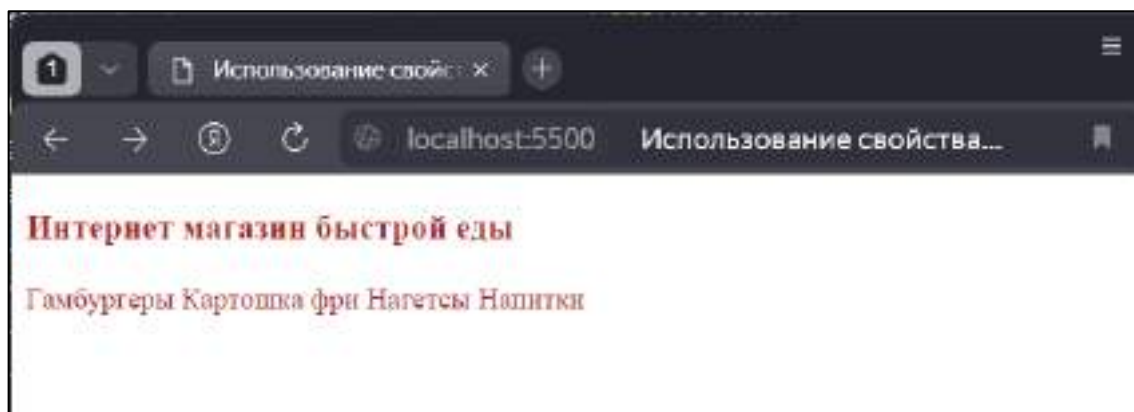


Рисунок 6.33 - Использование свойства display со значением inline

3) Использование свойства display со значением inline-block. Еще одно значение - inline-block - представляет элемент, который обладает смесью признаков блочного и строчного элементов. По отношению к соседним внешним элементам такой элемент расценивается как строчный. То есть он не отделяется от соседних элементов переводом строки. Однако по отношению к вложенным элементам он рассматривается как блочный. И к такому элементу применяются свойства width, height, margin.

```
<head> <style>
```

```
span{ width: 100px; height: 30px; background-color: LightSeaGreen; }
```

```
.inlineBlockSpan{ display: inline-block;} </style> </head>
```

```
<body> <p> Хвойный лес <span> - лес</span>, состоящий из деревьев одной или  
нескольких хвойных пород: сосны, ели, пихты, лиственницы и другие.</p>
```

```
<p> Лиственный лес<span class="inlineBlockSpan">- лес</span>, состоящий  
из лиственных пород деревьев и кустарников </p> </body>
```

Реализация данного кода представлена на рисунке 6.34.

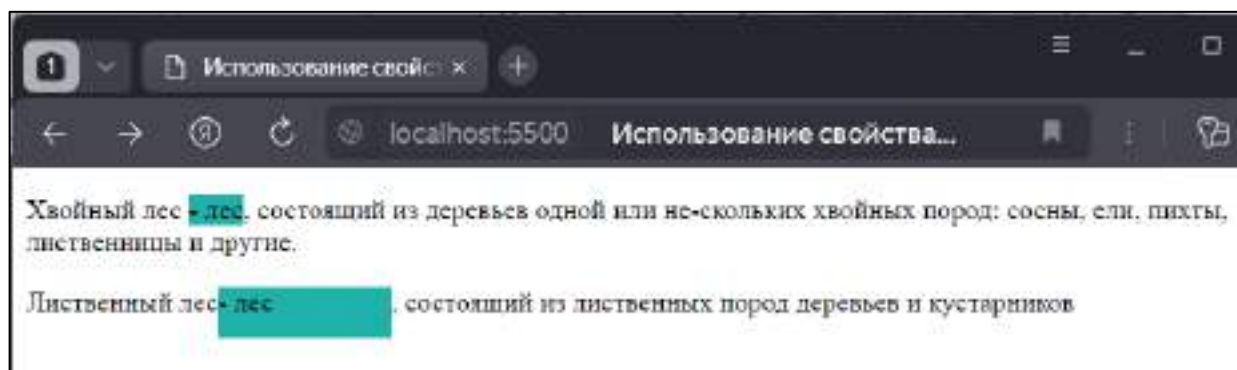


Рисунок 6.34 – Использование свойства display со значением inline-block

Первый элемент `span` является строчным, у него значение `inline`, поэтому для него бессмысленно применять свойства `width` и `height`. А вот второй элемент `span` имеет значение `inline-block`, поэтому к нему также применяются и ширина, и высота, и при необходимости еще можно установить отступы.

4) Использование свойства `display` со значением `run-in`. Значение `run-in` определяет элемент, который зависит от соседних элементов. И здесь есть три возможных варианта:

- Элемент окружен блочными элементами, тогда фактически он имеет стиль `display: block`, то есть сам становится блочным.
- Элемент окружен строчными элементами, тогда фактически он имеет стиль `display: inline`, то есть сам становится строчным.

Во всех остальных случаях элемент считается блочным.

5) Использование свойства `display` со значением `table`. Значение `table`, по сути, превращает элемент в таблицу. Для этого у элемента списка устанавливается стиль `display: table-cell`.

```
<head> <style>
ul { display: table; margin: 0; }
li { list-style-type: none; display: table-cell; padding: 10px; } </style> </head>
<body color: BlueViolet;> <h2>Корзина покупок:</h2>
<ul> <li>Товар</li> <li>Цена</li> <li>Количество</li>
<li>Печенье</li> <li>50 р.</li> <li>3 пачки</li>
<li>Чай</li> <li>100 р.</li> <li>1 пачка.</li>
<li>Конфеты</li> <li>700 р.</li> <li>1 кг.</li> </ul> </body>
```

Реализация данного кода представлена на рисунке 6.35.

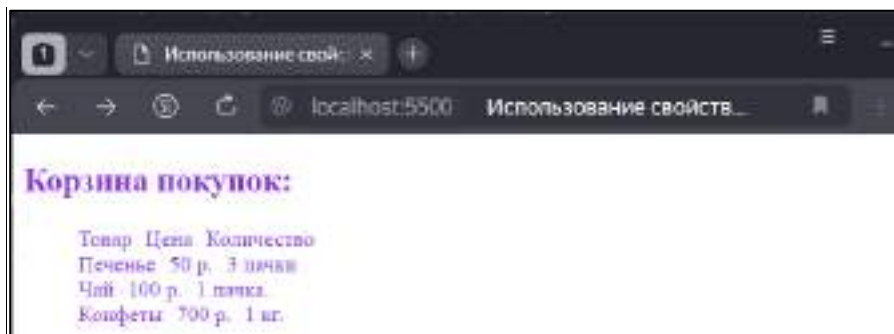


Рисунок 6.35 – Использование свойства `display` со значением `table`

б) Использование свойства `display` со значением `none`. Значение `none` позволяет скрыть элемент, которого как будто нет на веб-странице.

```
<head> <style>
.invisible { display: none; } </style> </head>
<body color: Teal;> <h2> Содержание романа “Капитанская дочка”</h2>
<p>Глава I. Сержант гвардии.</p>
<p>Глава II. Вожатый.</p>
<p>Глава III. Крепость.</p>
<p>Глава IV. Поединок.</p>
<p>Глава V. Любовь.</p>
<p class="invisible">Глава VI. Пугачевщина.</p>
<p>Глава VII. Приступ.</p> </body>
```

Реализация данного кода представлена на рисунке 6.36.

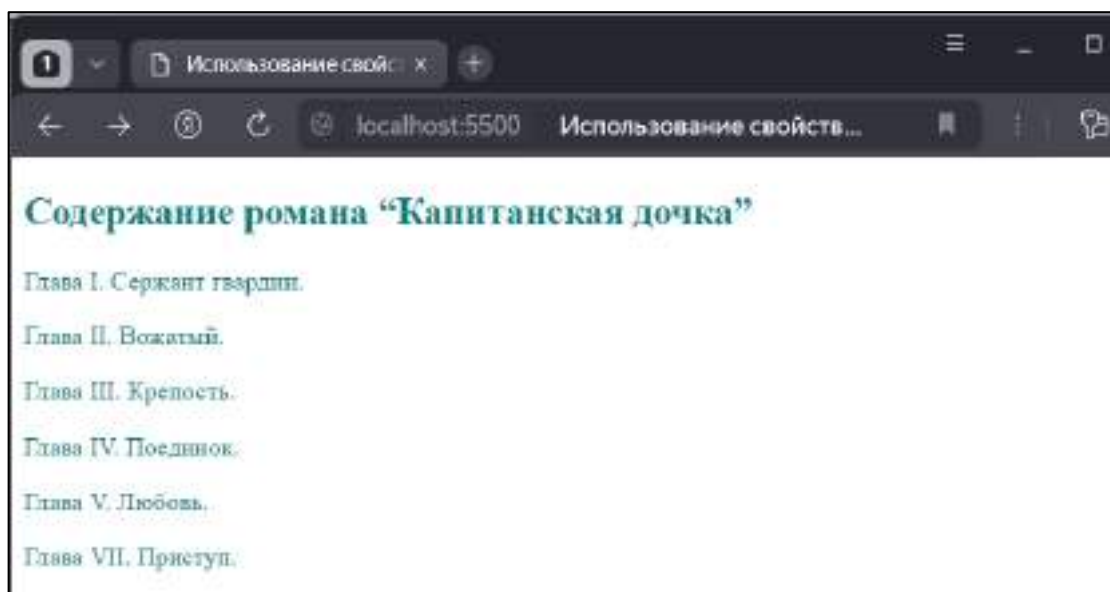


Рисунок 6.36 - Использование свойства `display` со значением `none`

6.6.5.5 Свойство `visibility`

Как следует из названия, этого свойство управляет видимостью блока. Оно может принимать только два значения:

- `visible` – обычное состояние блока (по умолчанию);
- `hidden` – блок становится прозрачным, т.е. невидимым.

Свойство `visibility:hidden` очень похоже на свойство `display:none`. Однако свойство `visibility:hidden` делает элемент прозрачным, но при этом продолжает присутствовать на странице, а свойство `display:none` отображается так, как будто его никогда не было. Далее представлен пример использования свойства `visibility`.

```
<head> <body color: SlateBlue;> <h2>Мир сказок.</h2><hr>
<h3> Сказки А.С. Пушкина:</h3>
  <p> Сказка о рыбаке и рыбке.</p>
  <p> Сказка о царе Салтане.</p>
  <p style="display:none"> Сказка о мёртвой царевне и о семи богатырях</p>
  <p> Сказка о золотом петушке</p>
  <p>Сказка о попе и о работнике его Балде</p><hr>
  <h3> Сказки В.И. Даля: </h3><hr>
  <p>Сказка “Девочка Снегурочка”. </p>
  <p> Сказка “Лучший певчий”. </p>
  <p style="visibility:hidden">Сказка “О дятле”.</p>
  <p>Сказка “Лучший певчий”. </p> </body>
```

Реализация данной HTML-страницы в браузере представлена на рисунке 6.37.

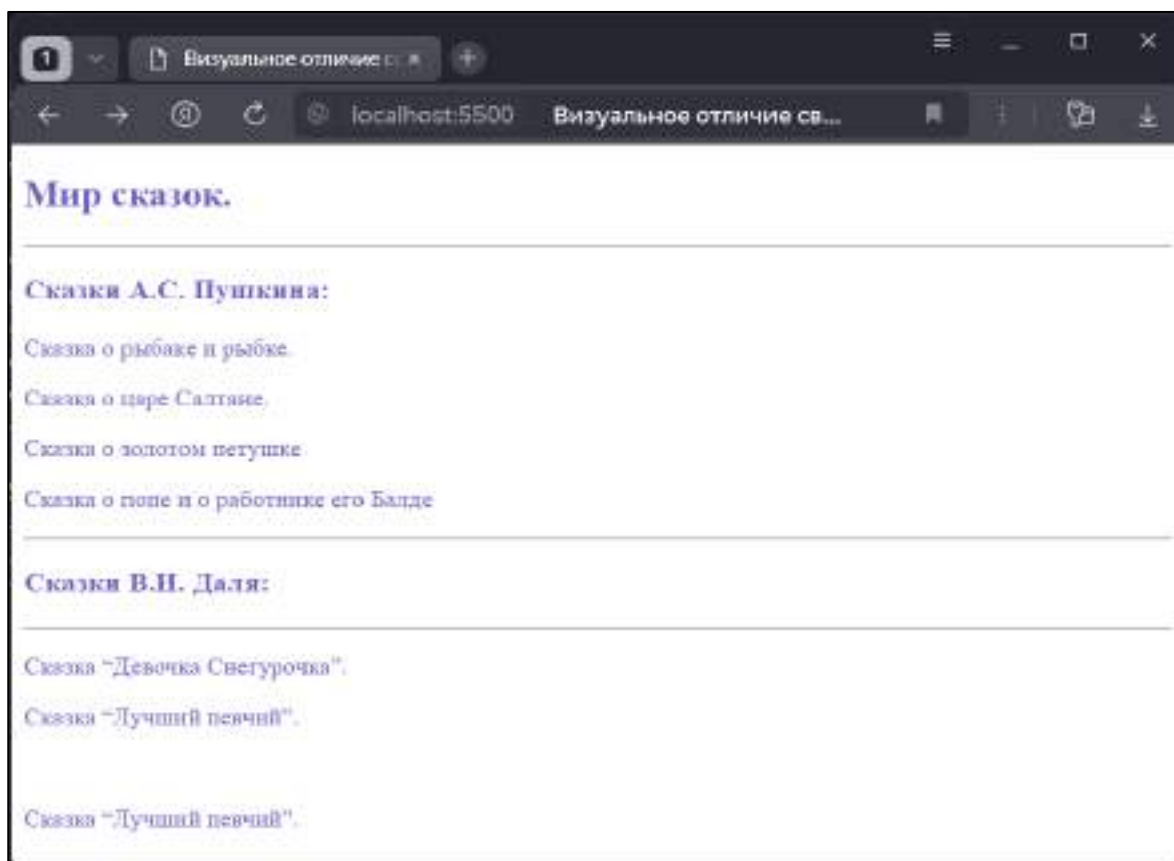


Рисунок 6.37 – Визуальное отличие свойств “`visibility:hidden`” и “`display:none`”

Как видно из рисунка 6.37, в первой группе второй абзац с установленным свойством “display:none” просто отсутствует на экране, поэтому сразу после первого абзаца идет третий. Во второй группе абзац с установленным свойством “visibility:hidden” также отсутствует на экране, но его место остается незанятым. Иначе говоря, он также остается на своем месте, но является прозрачным, вследствие чего невидимым. С помощью свойства visibility можно организовывать раскрывающиеся меню. В начальном состоянии видимость всех пунктов этого меню устанавливается как “visibility:hidden”. Когда пользователь щелкает по команде главного меню, то включается видимость нужного пункта, с помощью объявления “visibility:visible”.

6.7 Создание панели навигации

Панель навигации играет важную роль на сайте, поскольку позволяет перемещаться между страницами сайта или на внешние ресурсы. По сути, панель навигации представляет собой набор ссылок, часто в виде нумерованного списка. Панели навигации бывают разных форм: вертикальные и горизонтальные, одноуровневые и многоуровневые, но в каждом случае элемент <a> находится в центре каждой навигации. Поэтому при создании панели навигации вы можете столкнуться с рядом трудностей из-за ограничений элемента ссылки. А именно, элемент <a> является встроенным, что означает, что вы не можете указать для него ширину, высоту или отступы. Ширина ссылки автоматически занимает необходимое ей пространство

6.7.1 Вертикальное меню

Для создания вертикального меню необходимо ссылку блочным элементом с помощью значения свойства display: block.

```

<head>
  <style>    ul.nav{ margin-left: 1px; padding-left:1px;
              list-style: none;
              background-color: Lavender;
              border: 2px solid;
              border-color: MediumOrchid; }
  ul.nav a { display: block; width: 8em;
              padding:8px;
              background-color: Lavender;
              border: 2px solid;
              border-color: MediumOrchid;
              text-decoration: none; }
  ul.nav li:last-child a { border-bottom: 2px; } </style> </head>
  <body> <h2>Интернет-магазин женской обуви</h2>
  <ul class="nav">
    <li><a href="#">Сапоги</a></li>
    <li><a href="#">Ботильоны</a></li>
    <li><a href="#">Туфли</a></li>
    <li><a href="#">Лоферы </a></li>
    <li><a href="#">Босоножки</a></li>
  </ul> </body>

```

Реализация данного кода представлена на рисунке 6.38.

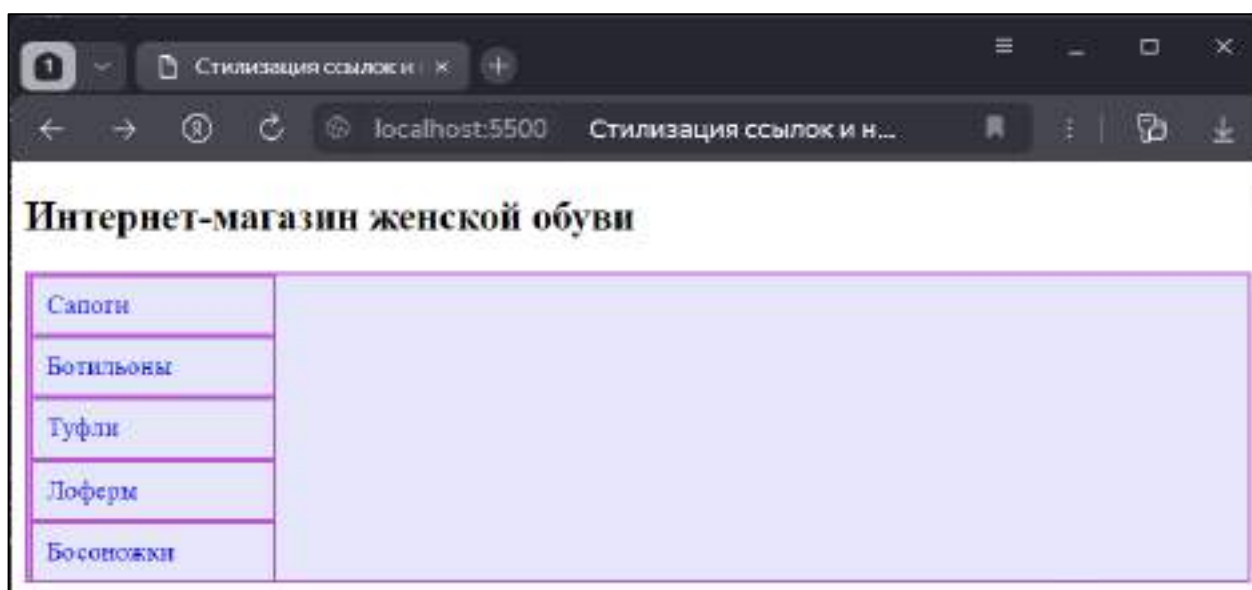


Рисунок 6.38 - Стилизация ссылок и навигация в CSS3

После установки свойства `display: block` можно определить у блока ссылки ширину, отступы и т.д.

6.7.2 Горизонтальное меню

Существует два метода создания горизонтального меню. Первый - использовать свойство `float` и создавать плавающие элементы из ссылок, которые обтекают друг друга слева. И второй способ - создать соединительную линию (строку ссылок), задав свойство `display:inline-block`.

6.7.2.1 Использование `float`

Алгоритм создания панели навигации с использованием `float` делится на два этапа. На первом этапе элемент `li`, содержащий ссылку, имеет значение `float: left;`. Это позволяет выровнять все элементы списка по ширине, так что правый элемент списка обтекает левый элемент списка.

Второй этап - настройка ссылки `display:block` на элемент, что дает возможность задавать ширину, отступы и вообще все особенности, характерные для блочных элементов.

```
<head> <style>    ul.nav{ margin-left: 1px; padding-left: 1px; list-style: none;}
    .nav li {float: left;}
    ul.nav a {display: block; width: 6em; padding:12px;
margin: 16px; background-color: LightCyan;
border: 2px; text-decoration: none; border-color: DarkTurquoise; text-align: center; }
    ul.nav a:hover{background-color: Cyan; border-color: DarkTurquoise; } </style> </head>
<body> <h2>Магазин сумок</h2>
    <ul class="nav"> <li><a href="#">Клатч</a></li>
        <li><a href="#">Рюкзак</a></li>
        <li><a href="#">Офисная сумка</a></li>
        <li><a href="#">Пляжная сумка</a></li> </ul> </body>
```

Реализация данного кода представлена на рисунке 6.39.

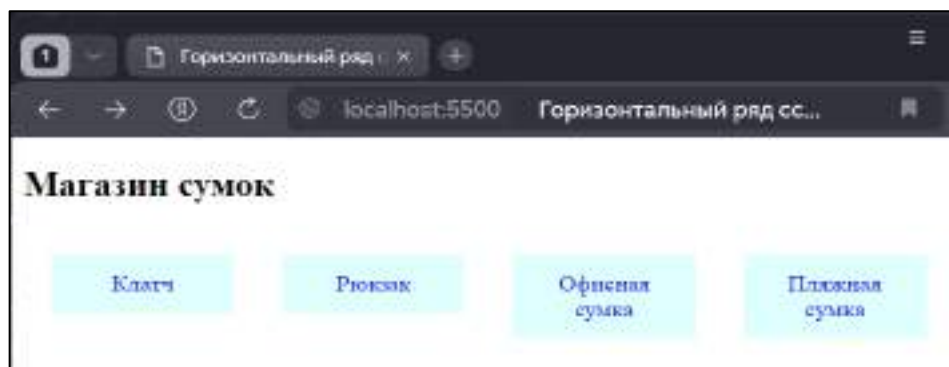


Рисунок 6.39 - Горизонтальный ряд ссылок в CSS3

6.7.2.2 Inline и inline-block

Чтобы создать горизонтальную панель навигации, нужно написать каждый элемент li строчным, т.е. установить для него `display:inline`. Затем для элемента link, который находится внутри элемента li, нужно установить `display: inline-block`.

```
<head> <style> ul.nav{ margin-left: 1px; padding-left: 1px; list-style: none; }
.nav li { display: inline; }
ul.nav a {display: inline-block; width: 6em; padding:12px;
background-color: MistyRose; border: 2px; border-color: MediumVioletRed;
text-decoration: none; text-align: center; }
ul.nav a:hover {background-color: HotPink; border-color:MediumVioletRed; border: 2px;}
</style> </head> <body> <h2> Интернет-магазин спортивной обуви</h2>
<ul class="nav"> <li><a href="#"> Кроссовки /a></li>
<li><a href="#">Кеды</a></li> <li><a href="#"> Слипоны</a></li>
<li><a href="#">Мокасины</a></li> </ul> </body>
```

Реализация данного кода представлена на рисунке 6.40.

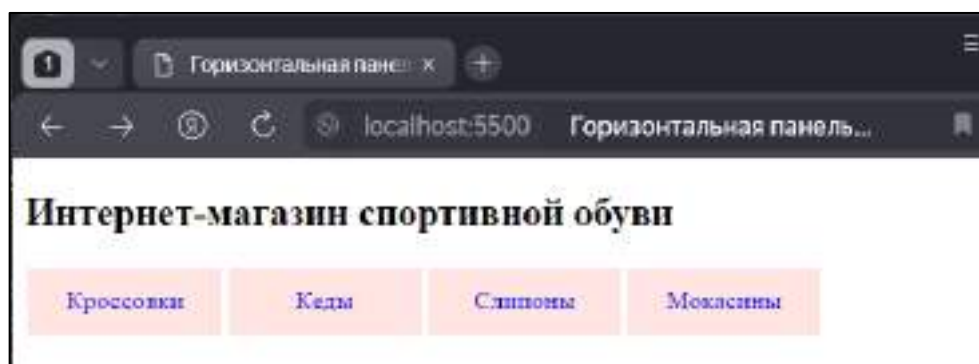


Рисунок 6.40 - Горизонтальная панель навигации в CSS3

6.8 Использование свойства box-sizing

При работе с плавающими элементами и свойством float может возникнуть распространенная проблема: плавающие элементы выпадают со страницы. Рассмотрим наглядно эту проблему на конкретном примере.

```
<head> <style> #sidebar {float: left; width: 25%; padding: 10px; color: BlueViolet; }  
#main{ border-left: 1px solid #ccc; width:75%; padding: 15px;  
margin-left: 25%; color: DarkViolet; } </style> </head>
```

```
<body> <div id="sidebar"><h2>Новости театра музыкальной комедии города  
Оренбурга</h2>
```

```
<p>В оренбургском театре музыкальной комедии в феврале и марте пройдут спектакли  
мюзикла “Ромео и Джульетта”, основанные на трагедии, написанной Уильямом Шекспиром  
в 1594 году. В ней рассказывается о любви юноши и девушки из двух враждующих веронских  
родов - Монтекки и Капулетти.</p> </div>
```

```
<div id="main"> <h2>О мюзикле Ромео и Джульетта</h2>
```

```
<p> Действие спектакля разворачивается в Вероне, где непримиримо враждуют два  
рода – Монтекки и Капулетти. Однако любовь - искренняя и настоящая - всегда находит путь  
к сердцу. Так и случилось – юный Ромео влюбляется в прекрасную Джульетту. Мюзикл  
рассказывает о том, что кровная вражда, взаимная ненависть, столкновение мировоззрений  
всегда будут бессильны перед лицом истинной бесконечной Любви. Так было много лет назад,  
так происходит и сейчас. История о Ромео и Джульетте особенно актуальна “в наше скупое на  
искренние чувства время”.</p> </div></body>
```

Реализация данного кода представлена на рисунке 6.41.

Как видно на рисунке 6.41 буквы вылезают из плавающего блока за границу. Это обусловлено тем, что для свойства box-sizing по умолчанию используется значение content-box, то есть при определении ширины и высоты элемента браузер будет прибавлять к значению свойств width и height также и внутренние отступы padding и ширину границы. В итоге это может привести к выпадению плавающих элементов из тех блоков, которые для них предназначены. Рекомендуется устанавливать для свойства box-sizing значение border-box, чтобы все элементы измерялись одинаково, а их ширина представляла только значение свойства width.

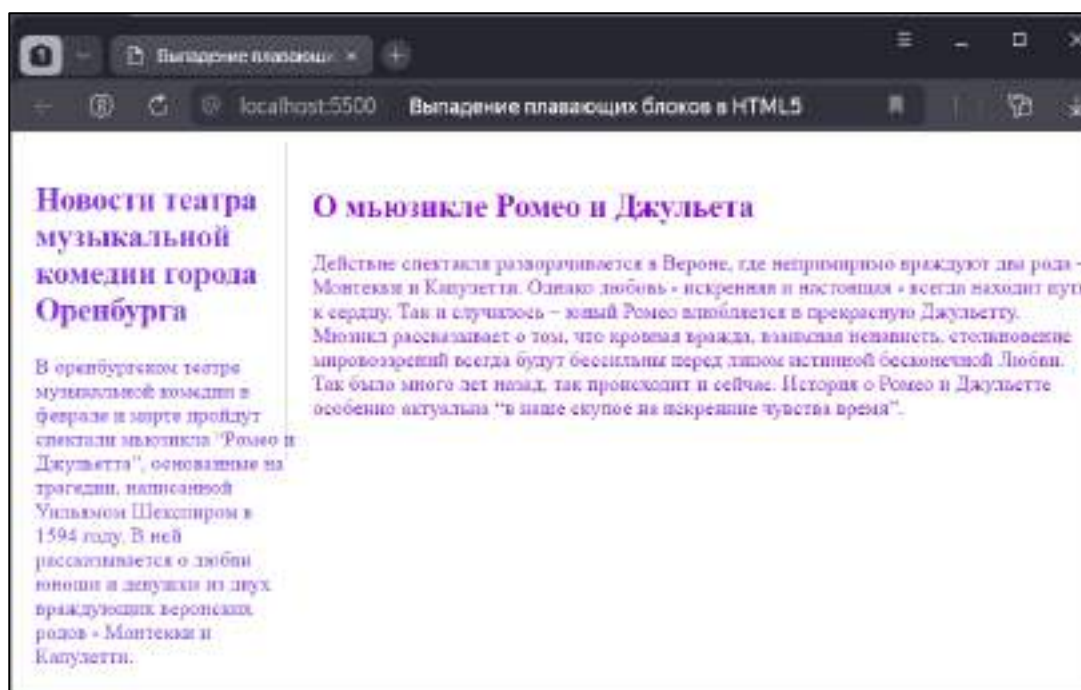


Рисунок 6.41 - Выпадение плавающих блоков в HTML5

Поэтому нередко в стилях добавляется следующий стиль.

*** {box-sizing: border-box;}**

То есть значение `box-sizing: border-box;` устанавливается для всех элементов, и все они интерпретируются браузером одинаково. К примеру, если добавить этот стиль в выше определенную страницу, получится другой результат (рисунок 6.42).

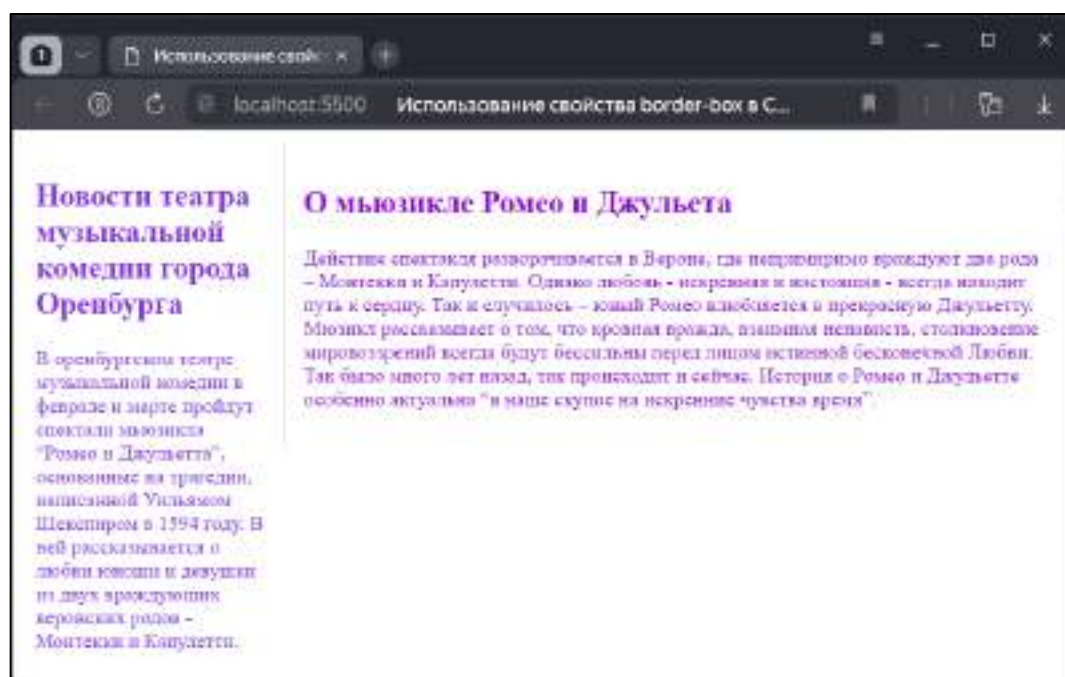


Рисунок 6.42 – Использование свойства border-box в CSS3

6.9 Создание макета веб-страницы из блоковых элементов

На основе информации, полученной в предыдущих темах, далее представлено описание создания макета простейшей веб-страницы. На первых этапах определяется базовая структура веб-страницы (рисунок 6.43).

```
<head> <link href="styles.css" rel="stylesheet">
  <div id="header"> <h1> Туристическая компания “White Lily”</h1>
    <div id="nav">
      <ul><li><a href="#">Поиск туров</a></li>
        <li><a href="#">Поиск отелей</a></li>
        <li><a href="#">Горящие туры </a></li>
        <li><a href="#">Выбрать турагентство </a></li>
        <li><a href="#">О нас </a></li> </ul> </div>    </div>
    <div class="wrapper"> <div id="sidebar1" class="aside"> <h2>Направления туризма</h2>
      <p> Туристическая компания “White Lily”, являясь одним из признанных лидеров
        российского рынка путешествий и туризма, предлагает отдых в самых красивых
        местах Турции, России, Египта, Туниса, Греции, Индии, Таиланда, Индонезии, Кипра, Кубы,
        ОАЭ. Идет постоянная работа по открытию новых направлений. </p></div>
        <div id="sidebar2" class="aside"> <h2>Преимущества компании “White Lily”</h2>
          <p> Количество туристов, выбирающих отдых с “White Lily”, постоянно растет, а это
            лучшее подтверждение высококачественного обслуживания и профессионализма наших
            сотрудников. Мы подбираем самые популярные среди российских туристов отели всех
            ценовых категорий, обеспечиваем надежные и качественные авиаперевозки. </p>
          <div> <div id="article"> <h2> О туристической компании “White Lily”</h2>
            <p> Бренд “White Lily” представлен на рынке с 2001 года. Сегодня т компания
              занимает лидирующие позиции в туристической отрасли и позиционируется как марка
              надежности и качества. Компания организует групповые и индивидуальные туры на базе
              собственных чартерных программ и регулярных рейсов, занимается развитием конгресс-,
              спортивного и других видов туризма, а также активно продает авиабилеты в онлайн. Мы
              помогаем найти и приблизить счастье от путешествия вашей мечты. </p></div></div> <div
            id="footer">
              <p>Contacts: WhiteLily @mail.ru</p><p>Copyright © WhiteLily.com, 2024</p> </div></body>
```

Реализация данного кода представлена на рисунке 6.43.

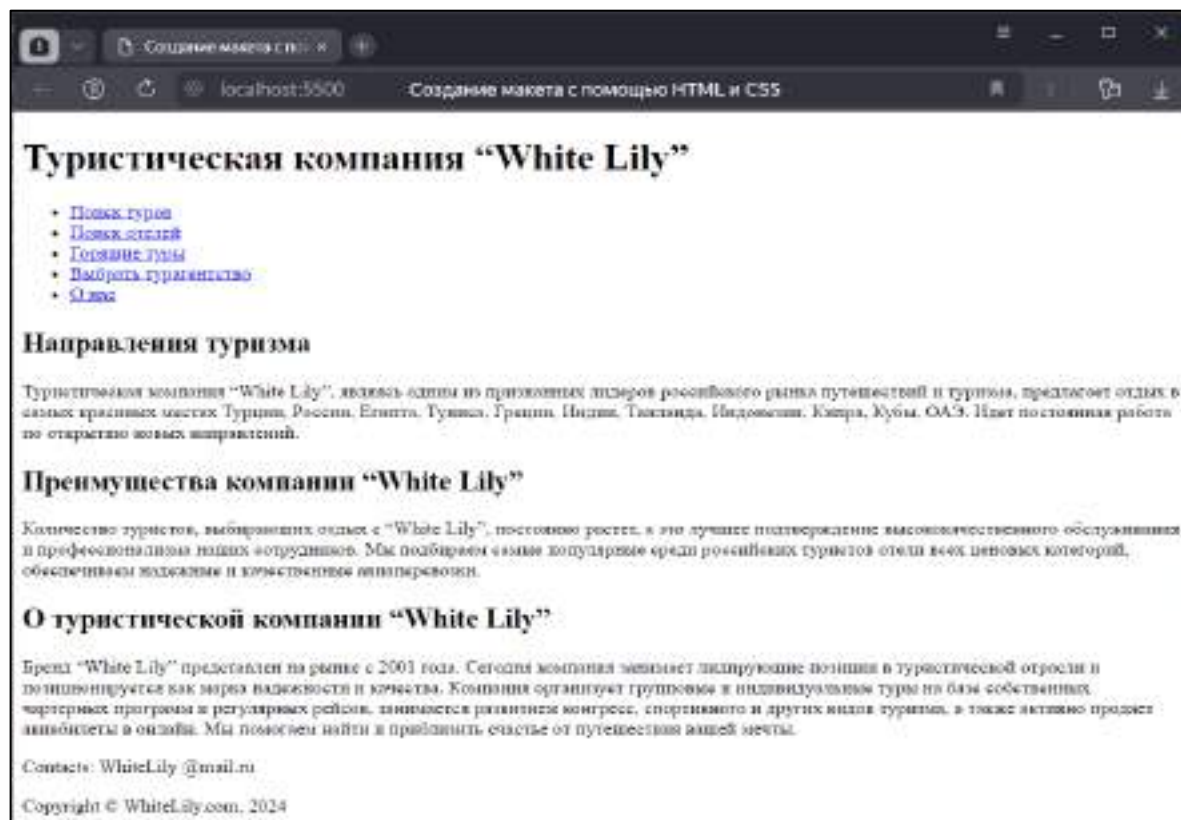


Рисунок 6.43 - Создание макета с помощью HTML и CSS

Вначале находится шапка сайта - блок с header, содержащий заголовок страницы и панель навигации. Далее идет блок wrapper, который содержит две боковые панели и блок основного контента страницы. Боковые панели также содержат некоторый контент, но главное, что они определены до основного блока. А в самом низу есть небольшой футер.

В начале веб-страницы определяется подключение файла styles.css, который будет стилизовать веб-страницу. Поэтому создается файл styles.css в том же каталоге, что и веб-страница, и определяется в нем следующее содержимое.

```
* { box-sizing: border-box; }
html, body, div, span, h1, h2, h3, h4, h5, h6, p, a, ul, li
{ margin: 0;
padding: 0; border: 0;
font-size: 100%;
vertical-align: baseline; }
```



```

body { font-family: Verdana,
Arial, sans-serif;
background-color: #bcfafa; }
#header { background-color: LightBlue;}
#header h1 { font-size: 24px; text-transform: uppercase; font-weight: bold; padding: 30px
30px 30px 10px; clear: both;}
#nav { background-color: LightSkyBlue;
border-top: 1px solid DarkTurquoise;
border-bottom: 1px solid DarkTurquoise; }
#nav li {float: left; list-style: none; }
#nav a { display: block;
color: black; padding: 10px 25px;
text-decoration: none;
border-right: 1px solid DarkTurquoise; }
#nav li:last-child a { border-right: none;}
#nav a:hover { font-weight: bold; }
#nav:after { content: " "; display: table; clear: both; }
.wrapper { background-color: #b4ebf6; }
.aside h2 { font-size: 0.95em; margin-top: 15px; }
.aside h3 { font-size: 0.85em; margin-top: 10px; }
.aside p, .aside li {font-size: .75em; margin-top: 10px; }
.aside li { list-style-type: none; }
#sidebar1 { float: left; width: 20%; padding: 0 10px 0 20px; }
#sidebar2 { float: right; width: 20%; padding: 0 20px 0 10px; }
#article { background-color: #9aeff1;
border-left: 1px solid Cyan;
border-right: 1px solid Cyan;
margin-left: 20%; margin-right: 20%; padding: 15px; width: 60%; }
#article:after { clear: both; display: table; content: " "; }
#article h2 { font-size: 1.3em; margin-bottom: 15px; }
#article p { line-height: 150%; margin-bottom: 15px; }
#footer { border-top: 1px solid Azure; font-size: .8em; text-align: center; padding: 10px
10px 30px 10px; }
#nav ul, #header h1, .wrapper, #footer p { max-width: 1200px; margin: 0 auto; }
.wrapper, #nav, #header, #footer { min-width: 768px;}

```

Реализация данного кода представлена на рисунке 6.44.

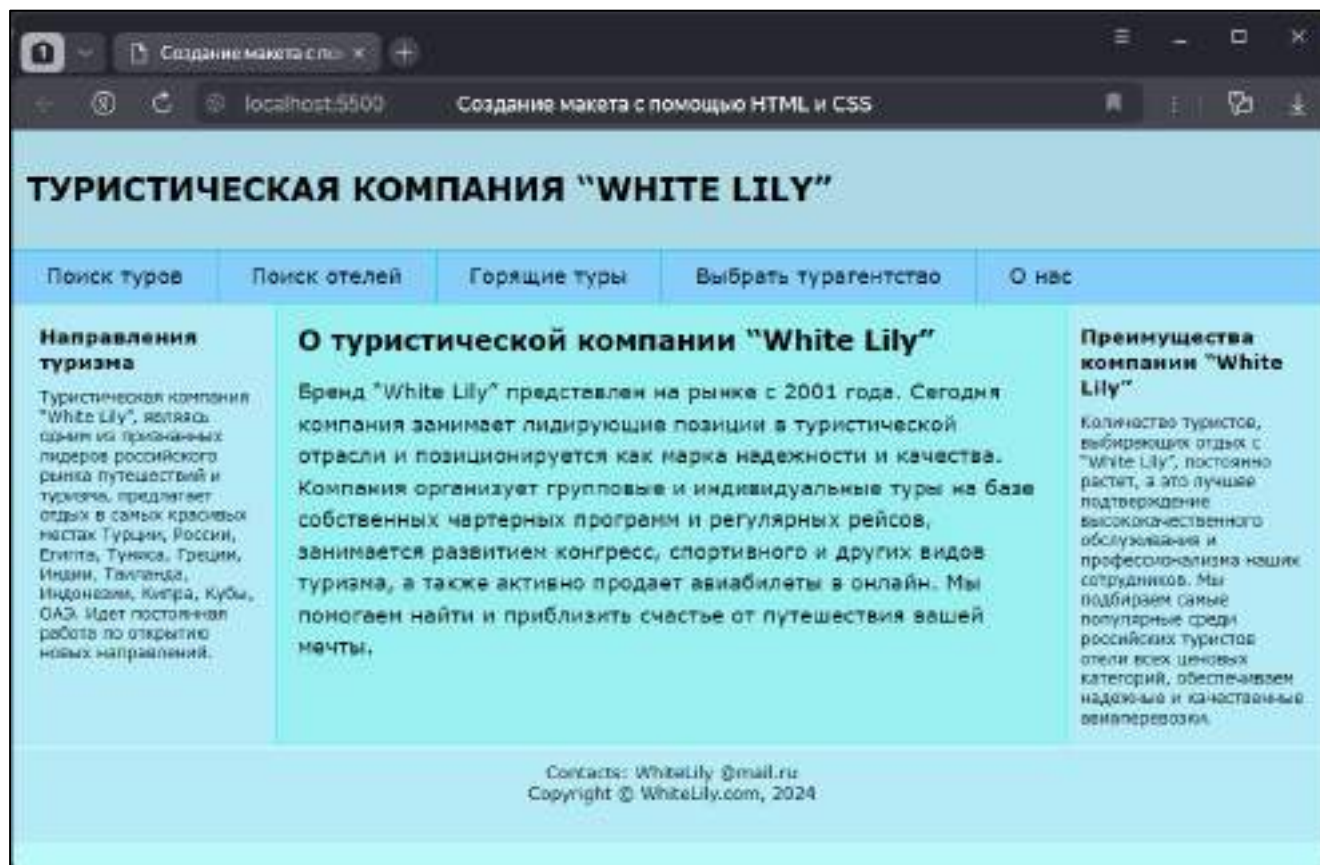


Рисунок 6.44 - Итоговое форматирование веб-страницы

Первые три стиля сбрасывают настройки стиля по умолчанию для используемых элементов, а также устанавливают стиль элемента `body`.

Следующая пара стилей управляет отображением заголовка и заголовка страницы:

```
#header { background-color: LightBlue;}
```

```
#header h1 { font-size: 24px; text-transform: uppercase; font-weight: bold; padding: 30px 30px 30px 10px; clear: both;}
```

Следующий набор стилей управляет созданием горизонтальной панели:

```
#nav { background-color: LightSkyBlue; border-top: 1px solid DarkTurquoise; border-bottom: 1px solid DarkTurquoise; }
```

```
#nav li {float: left; list-style: none; }
```

```
#nav a { display: block; color: black; padding: 10px 25px; text-decoration: none; border-right: 1px solid DarkTurquoise; }
```

```
#nav li:last-child a { border-right: none;}
```

```
#nav a:hover { font-weight: bold; }
```

```
#nav:after { content: " "; display: table; clear: both; }
```

Далее создается горизонтальная панель навигации: элементам `` присваивается `float (float: left;)`, который размещает их в ряд, и каждая ссылка становится блочным элементом (`display: block;`). Далее настраивается средняя часть страницы, особенно боковые панели:

```
.wrapper { background-color: #b4ebf6; }
```

```
.aside h2 { font-size: 0.95em; margin-top: 15px; }
```

```
.aside h3 { font-size: 0.85em; margin-top: 10px; }
```

```
.aside p, .aside li {font-size: .75em; margin-top: 10px; } .aside li { list-style-type: none; }
```

```
#sidebar1 { float: left; width: 20%; padding: 0 10px 0 20px; }
```

```
#sidebar2 { float: right; width: 20%; padding: 0 20px 0 10px; }
```

Стиль класса `wrapper` позволяет установить цвет фона для боковых панелей. Для каждой боковой панели ширина установлена на уровне 20% ширины страницы. Проценты позволяют автоматически подстраивать ширину блоков под ширину окна браузера при его расширении или сжатии.

Ниже приведены стили для основного блока контента и нижнего колонтитула:

```
#article { background-color: #9aeff1; border-left: 1px solid Cyan; border-right: 1px solid Cyan; margin-left: 20%; margin-right: 20%; padding: 15px; width: 60%; }
```

```
#article:after { clear: both; display: table; content: " "; }
```

```
#article h2 { font-size: 1.3em; margin-bottom: 15px; }
```

```
#article p { line-height: 150%; margin-bottom: 15px; }
```

```
#footer { border-top: 1px solid Azure; font-size: .8em; text-align: center; padding: 10px 10px 30px 10px; }
```

Поскольку боковые панели имеют ширину в 20% каждая, то для главного блока устанавливается ширина в 60% и отступы справа и слева в 20%.

И в конце идет пара довольно важных стилей:

```
#nav ul, #header h1, .wrapper, #footer p { max-width: 1200px; margin: 0 auto; }
```

```
.wrapper, #nav, #header, #footer { min-width: 768px; }
```

Вначале для ряда селекторов определена максимальная ширина 1200 пикселей. Это означает, что основные элементы страницы не будут выходить за пределы 1200 пикселей. А автоматические левые и правые поля позволят центрировать содержимое

элементов. При ширине браузера 1400 пикселей эти элементы шириной 1200 пикселей будут располагаться как бы посередине, а справа и слева будут отступы шириной $(1400-1200)/2=100$ пикселей. Второй стиль позволит создать фиксированную минимальную ширину для ряда элементов.

6.10 Контрольные вопросы

- 1) Перечислить свойства блочных элементов – margin, padding, border.
- 2) Задание фона в блочных элементах, описать различные варианты задания фона. Описать свойства background-color, background-image, background-size и др.
- 3) Позиционирование блочных элементов. Абсолютное, относительное, статическое, фиксированное позиционирование.
- 4) Статическое позиционирование. Использование свойства overflow.
- 5) Статическое позиционирование. Примеры работы со свойством float.

6.11 Тестирование по главе

- 1) Какое свойство блочных элементов задает внешний отступ, то есть расстояние от границы текущего элемента до других соседних элементов или до границ внешнего контейнера?
 - а) margin;
 - б) padding;
 - в) border;
 - г) content.

2) Какое свойство блочных элементов определяет внутренний отступ, определяет расстояние от границы элемента до внутреннего содержимого?

- a) margin;
- б) padding;
- в) border;
- г) content.

3) При работе с блочными элементами какое свойство устанавливает режим повторения фонового изображения по всей поверхности элемента?

- a) background-color;
- б) background-image;
- в) background-repeat;
- г) background-clip.

4) При работе с блочными элементами какое свойство определяет область, которая вырезается из изображения и используется в качестве фона?

- a) background-size;
- б) background-image;
- в) background-attachment;
- г) background-clip.

5) Какой вид позиционирования позволяет зафиксировать блок, независимо от прокрутки веб-страницы?

- a) absolute;
- б) relative;
- в) fixed;
- г) static.

6) Значение свойства position ... позиционирует элемент относительно границ элемента-контейнера

- a) static;
- б) absolute;
- в) relative;
- г) fixed.

7) Свойство ... позволяет изменить порядок следования элементов, при их наложении

- a) position;
- б) z-index;
- в) opacity;
- г) display.

8) Какое значение свойства display при статическом позиционировании позволяет преобразовать блочный элемент в строчный?

- a) inline;
- б) block;
- в) inline-block;
- г) list-item.

9) Какое значение свойства display при статическом позиционировании позволяет преобразовать блочный элемент?

- a) inline;
- б) block;
- в) inline-block;
- г) list-item.

10) Какое значение используют браузеры по умолчанию для box-sizing?

- a) content-box;
- б) border-box;
- в) content;
- г) border.

7 Трансформации, переходы и анимации

7.1 Трансформации

Одним из нововведений CSS3 по сравнению с предыдущей версией является встроенная возможность трансформации элемента. Преобразования включают в себя такие действия, как вращение элемента, его масштабирование, наклон или перемещение по вертикали или горизонтали. Чтобы создать преобразования в CSS3, используйте свойство `transform`.

7.1.1 Вращение

Для поворота элемента свойство `transform` использует функцию `rotate`:

`transform: rotate(угол_поворота deg)`

После слова `rotate` в скобках идет величина угла поворота в градусах. Например, блок поворачивается на 35 градусов.

Далее приведен пример использования вращения.

```
<head> <style>
    div {background-color: #32CD32;
    width: 100px; height: 100px; margin:6px; padding: 40px 15px;
    box-sizing: border-box;
    border: 2px solid black;
    display: inline-block; }
    .rotate{ transform: rotate(35deg); }
</style> </head>

<body style="margin-top: 20px;">
    <div> Данила Козловский </div>
    <div class="rotate">Александр Петров </div>
    <div> Сергей Безруков</div> </body>
```

Реализация данного кода представлена на рисунке 7.1.

Можно отметить, что при повороте вращаемый элемент может перекрывать соседние элементы, так как сначала задается положение элементов и только потом происходит поворот.

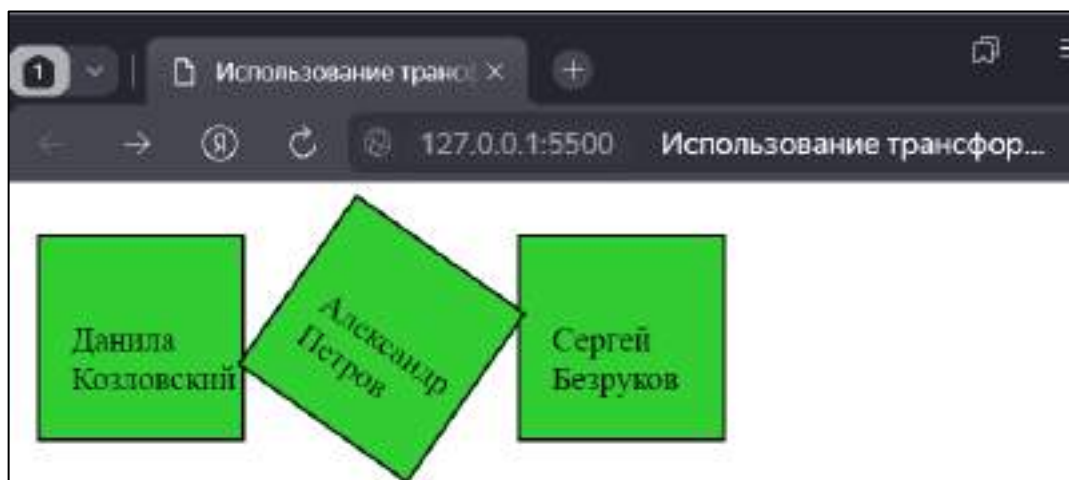


Рисунок 7.1 – Использование поворотов в блоковых элементах

Угол поворота может быть как положительным, так и отрицательным. В случае отрицательного значения вращение производится в противоположную сторону.

7.1.2 Масштабирование

Применение масштабирования имеет следующую форму:

transform: scale(величина_масштабирования)

Далее приведен пример использования масштабирования.

```
<head> <style> div { background-color: aqua; width: 100px; height: 100px; margin: 6px;
padding: 45px 15px; box-sizing: border-box;
border: 2px solid black; display: inline-block;}
.half_Scale{transform: scale(0.5);}
.double_Scale{transform: scale(2);}
</style> </head>
<body style="margin-top: 50px;" >
  <div>2020 год</div><div class="double_Scale">2021 год</div>
  <div>2022 год</div> <div class="half_Scale">2023 год</div>
</body>
```


Реализация данного кода представлена на рисунке 7.2.

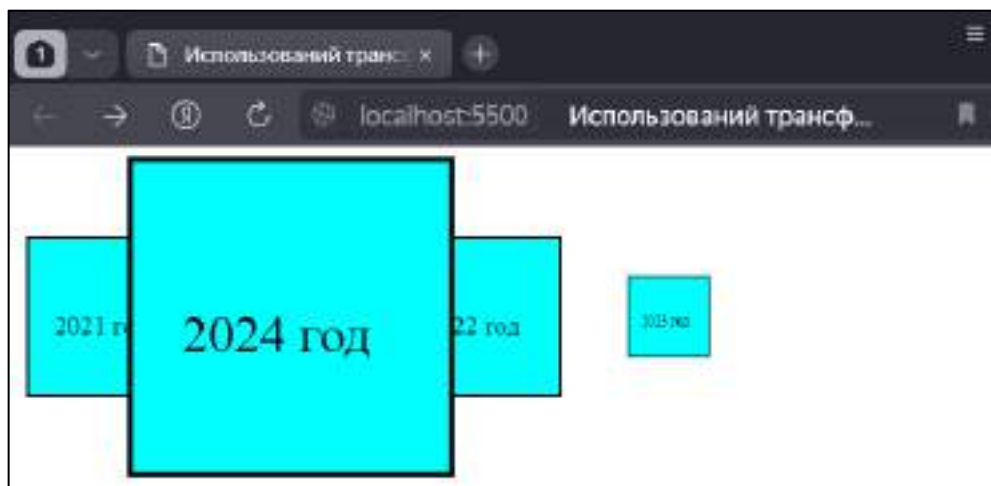


Рисунок 7.2 – Применение масштабирования в CSS

Значение больше 1 приводит к вертикальному и горизонтальному растяжению, а значение меньше 1 - к сжатию. То есть значение 0,5 приводит к уменьшению в два раза, а значение 1,5 - к увеличению в полтора раза.

Также можно установить значения масштаба отдельно для вертикали и горизонтали, как в нижеследующем примере.

```
<head> <style>
  div { background-color: purple;
    width: 130px;
    height: 130px;
    margin: 6px;
    padding: 45px 15px;
    box-sizing: border-box;
    border: 2px solid black;
    display: inline-block; }
  .scale { transform: scale(2, 0.5); }
</style> </head>
<body>
  <div>Москва</div>
  <div class="scale">Оренбург</div>
  <div>Санкт-Петербург</div>
</body>
```

Реализация данного кода представлена на рисунке 7.3.

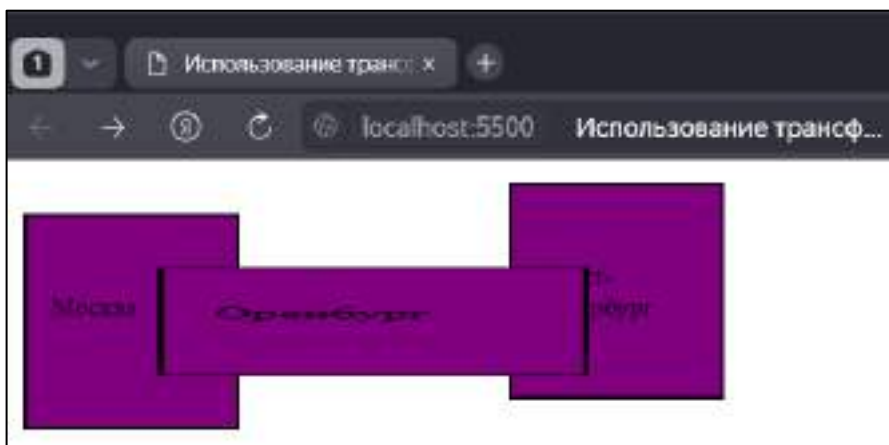


Рисунок 7.3 – Горизонтальное масштабирование в CSS

В данном случае по горизонтали будет идти масштабирование в 2 раза, а по вертикали - в 0.5 раз. Также можно по отдельности задать параметры масштабирования: функция `scaleX()` задает изменение по горизонтали, а `scaleY()` - по вертикали. Например:

`.scale{transform: scaleX(2);}`

Используя отрицательные значения, можно создать эффект зеркального отражения. Далее представлен пример применения такого эффекта.

```
<head>    div { background-color: coral;
width: 130px;
height: 130px;
margin:6px;
padding: 45px 15px;
box-sizing: border-box;
border: 2px solid black;
display: inline-block; }
.scale { transform: scaleX(-1); }
</style> </head>
<body>  <div>Телевизор</div>
<div class="scale">Холодильник</div>
<div>Пылесос</div> </body>
```

Реализация данного кода представлена на рисунке 7.4.

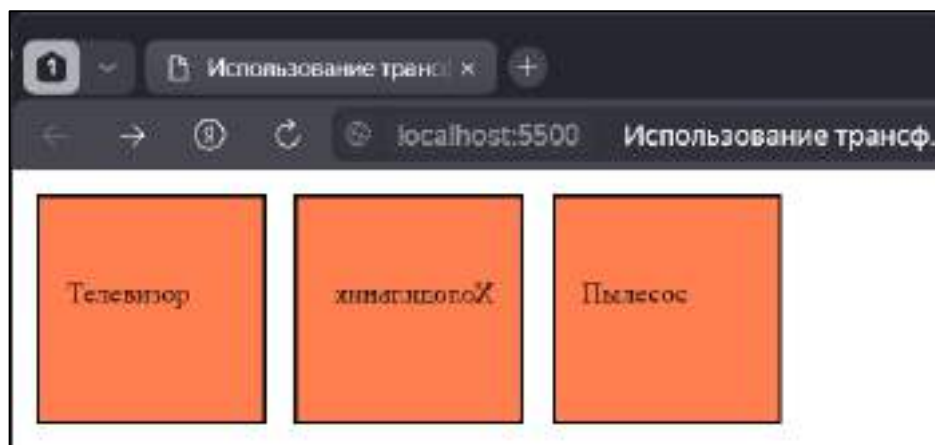


Рисунок 7.4 - Эффект зеркального отражения

7.1.3 Перемещение

Для перемещения элемента используется функция `translate`:

`transform: translate(offset_X, offset_Y)`

Значение `offset_X` указывает, на сколько элемент смещается по горизонтали, а `offset_Y` - по вертикали. Например, в нижеприведенном примере блок сместился на 30 пикселей вниз и на 50 пикселей вправо.

```
<head> <style>
img { background-color: silver;
width: 130px;
height: 130px;
margin: 6px;
box-sizing: border-box;
border: 2px solid black;
display: inline-block; }
.translate { transform: translate(50px, 30px);
background-color: blue; } </style>
</head> <body>


 </body>
```

Реализация данного кода представлена на рисунке 7.5.

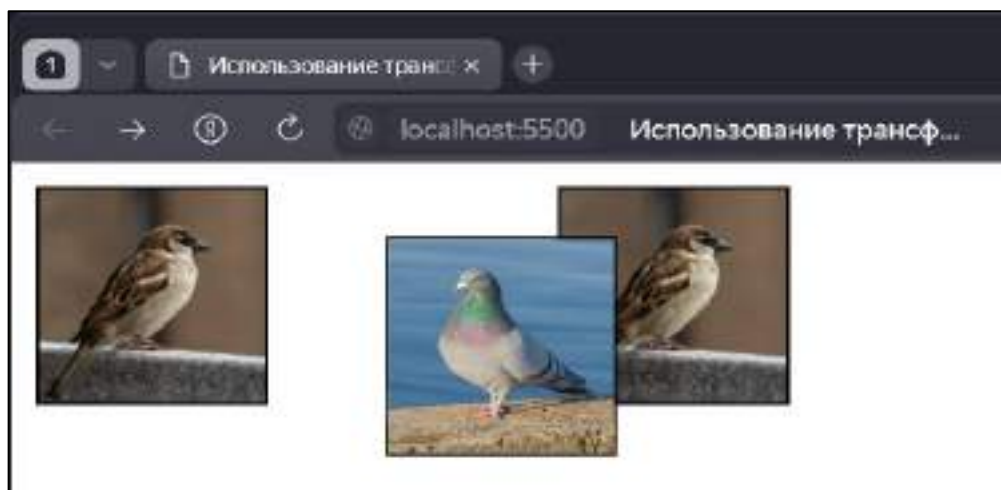


Рисунок 7.5 – Применение перемещения в CSS

В качестве единиц измерения смещения можно применять не только пиксели, но и любые другие единицы измерения длины в CSS - em, % и тд.

С помощью дополнительных функций можно отдельно применять смещения к горизонтали или вертикали: `translateX()` (перемещение по горизонтали) и `translateY()` (перемещение по вертикали). Например:

`transform: translateX(30px);`

Кроме положительных значений также можно использовать и отрицательные - они перемещают элемент в противоположную сторону:

`transform: translateY(-2.5em);`

7.1.4 Наклон

Для наклона элемента применяется функция `skew()`:

`transform: skew(X, Y);`

Первый параметр указывает, на сколько градусов наклонять элемент по оси X, а второй - значение наклона по оси Y.

Далее приведен пример использования наклонов.

```
<head> <style>
```

```
img{ background-color: silver;
```

```
width: 130px;
```

```

height: 130px;
margin: 6px;
box-sizing: border-box;
border: 2px solid black;
display: inline-block; }
.skewe { transform: skew(20deg, 5deg);
background-color: aqua; } </style> </head>

```

```

<body> 

 </body>

```

Реализация данного кода представлена на рисунке 7.6.

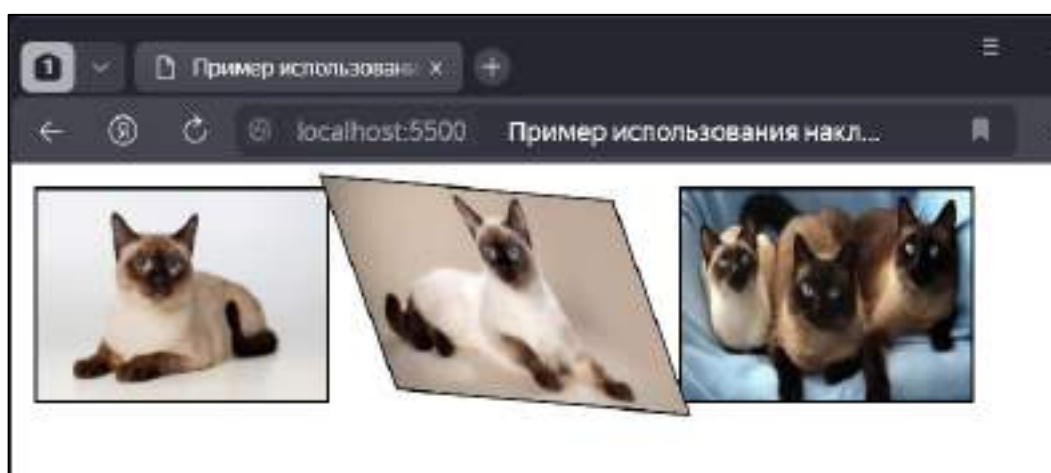


Рисунок 7.6 – Пример использования наклонов в CSS

Для создания наклона только по одной оси для другой оси надо использовать значение 0. Например, наклон на 45 градусов по оси X:

```
transform: skew(45deg, 0);
```

Или наклон на 45 градусов только по оси Y:

```
transform: skew(0, 45deg);
```

Для создания наклона отдельно по оси X и по оси Y в CSS есть специальные функции: `skewX()` и `skewY()` соответственно.

```
transform: skewX(45deg);
```

Также можно передавать отрицательные значения. Тогда наклон будет осуществляться в противоположную сторону:

transform: skewX(-30deg);

7.1.5 Комбинирование преобразований

Если надо применить к элементу сразу несколько преобразований, вращение и перемещение, то можно их комбинировать. Например, применение всех четырех преобразований:

transform: translate(50px, 100px) skew(30deg, 10deg) scale(1.5) rotate(90deg);

Браузер применяет все эти функции в порядке их следования. То есть в данном случае сначала к элементу применяется перемещение, потом наклон, потом масштабирование и в конце вращение.

7.1.6 Изменение исходной точки трансформации

По умолчанию при применении трансформаций браузер использует центр элемента в качестве отправной точки преобразования. Но с помощью свойства `transform-origin` можно изменить начало. Это свойство принимает значения в пикселях, `em` и процентах. Также можно использовать ключевые слова, чтобы поставить точку:

- `left top`: верхний левый угол элемента;
- `left bottom`: нижний левый угол элемента;
- `right top`: верхний правый угол элемента;
- `right bottom`: нижний правый угол элемента.

Далее представлен пример изменения исходной точки трансформации.

```
<style> div {background-color: #ccc; width: 100px; height: 100px;
margin: 80px 30px; float: left; box-sizing: border-box; border: 1px solid #333;}
.transform1 {transform: rotate(-45deg);}
.transform2 {transform-origin: left top; transform: rotate(-45deg);}
.transform3 {transform-origin: right bottom; transform: rotate(-45deg);} </style></head>
<body> <div class="transform1"></div>
      <div class="transform2"></div>
```

```
<div class="transform3"></div> </body>
```

Реализация данного кода представлена на рисунке 7.7.

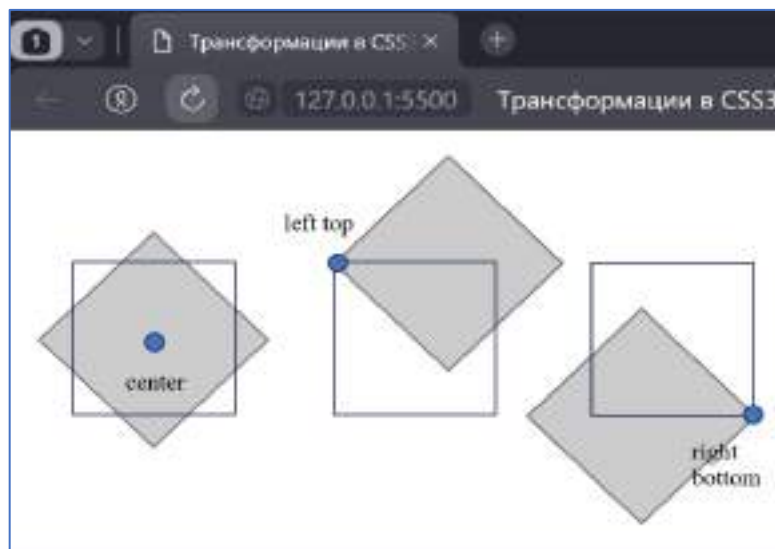


Рисунок 7.7 – Пример изменения исходной точки трансформации

7.2 Переходы

Переход (transition) представляет анимацию от одного стиля к другому в течение определенного периода времени. Чтобы создать переход, сначала необходимы два набора свойств CSS: начальный стиль, который будет иметь элемент в начале перехода, и конечный стиль, который будет результатом перехода.

Далее представлен пример использования переходов.

```
<head> <style>
  div {width: 100px; height: 100px; margin: 40px 30px;
    border: 1px solid #333; background-color: #ccc;
    transition-property: background-color; transition-duration: 2s;}
  div: hover {background-color: red;}
</style></head>
<body> <div></div>
</body>
```

Реализация данного кода представлена на рисунках 7.8 и 7.9.

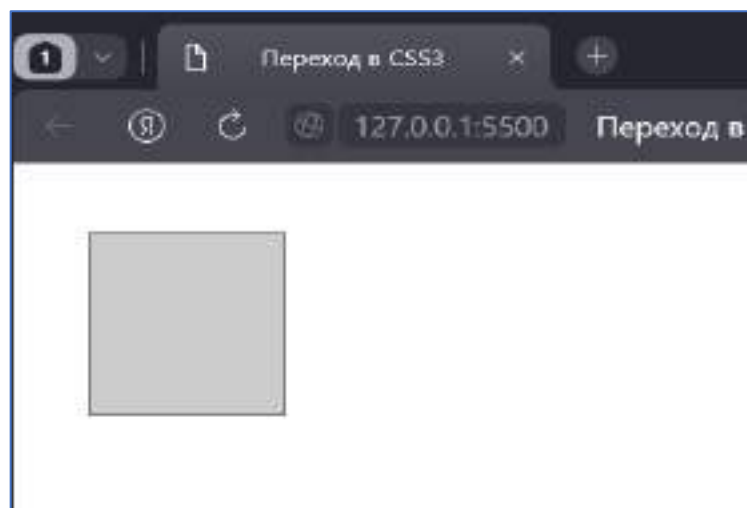


Рисунок 7.8 - Переход до

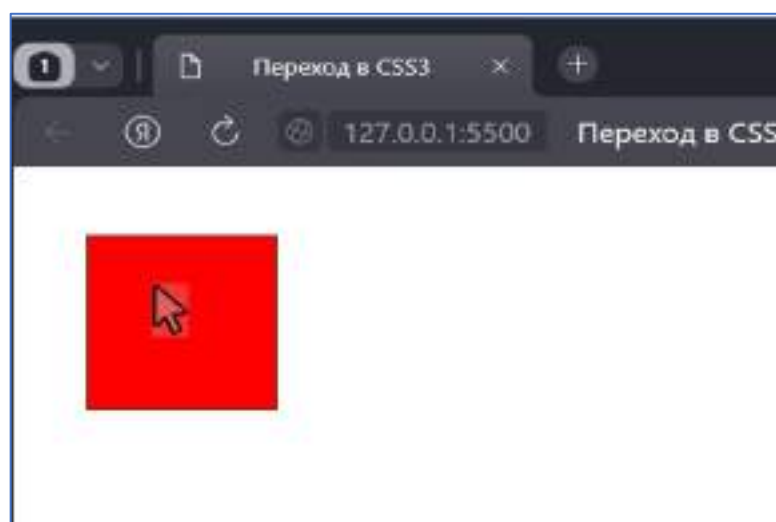


Рисунок 7.9 - Переход после

В данном примере анимируется свойство `background-color` элемента `div`. При наведении курсора мыши на элемент его цвет меняется с серого на красный. А когда указатель мыши отойдет от пространства элемента, вернется первоначальный цвет.

Чтобы указать свойство как анимируемое, его имя передается свойству `transition-property`.

`transition-property: background-color;`

Далее идет установка времени перехода в секундах с помощью свойства `transition-duration`:

`transition-duration: 2s;`

Помимо секунд можно устанавливать значения в миллисекундах, например, 500 миллисекунд:

transition-duration: 500ms;

Наконец, нужно определить инициатор действия и конечное значение свойства анимированного background-color. Инициатор представляет действие, которое приводит к смене одного стиля на другой. В CSS можно использовать псевдоклассы для запуска перехода. Например, здесь стиль псевдокласса: hover используется для создания перехода. То есть при наведении курсора мыши на элемент div сработает переход.

Помимо псевдокласса :hover можно использовать и другие псевдоклассы, например, :active (ссылка нажимается) или :focus (элемент, получающий фокус).

7.2.1 Переходы нескольких свойств

При необходимости можно анимировать сразу несколько свойств CSS. Так, в вышеприведенном примере изменим стили следующим образом:

```
div {width: 100px; height: 100px; margin: 40px 30px; border: 1px solid #333;  
background-color: #ccc;  
transition-property: background-color, width, height, border-color;  
transition-duration: 2s;}
```

```
div:hover {background-color: red; width: 120px; height: 120px; border-color:  
blue;}
```

Здесь анимируются сразу четыре свойства. Причем анимация для них всех длится 2 секунды, но можно для каждого свойства задать свое время:

```
transition-property: background-color, width, height, border-color;  
transition-duration: 2s, 3s, 1s, 2s;
```

Подобно тому как в свойстве transition-property через запятую идет перечисление анимируемых свойств, в свойстве transition-duration идет перечисление через запятую временных периодов для анимации этих свойств. Причем

сопоставление времени определенному свойству идет по позиции, то есть свойство width будет анимироваться 3 секунды.

Кроме перечисления через запятую всех анимируемых свойств можно просто указать ключевое слово all:

transition-property: all;

transition-duration: 2s;

Теперь будут анимироваться все необходимые свойства, которые меняют значения в стиле для псевдокласса :hover.

7.2.2 Функции переходов

Свойства transition-timing-function позволяют контролировать скорость движения и выполнения анимации. Другими словами, это свойство отвечает за то, как и в какое время анимация будет ускоряться или замедляться.

Это свойство может принимать в качестве значения одну из функций:

- linear: функция линейного сглаживания, свойство изменяется равномерно с течением времени;

- ease: функция плавности, при которой анимация ускоряется к середине и замедляется к концу, обеспечивая более естественное изменение;

- ease-in: функция плавности, при которой вначале происходит только ускорение;

- ease-out: функция плавности, которая ускоряется только в конце анимации;

- ease-in-out: функция плавности, при которой анимация ускоряется к середине и замедляется к концу, обеспечивая более естественные изменения;

- cubic-bezier: для анимации используется кубическая функция Безье.

Далее приведен пример использования функции transition-timing-function со свойством ease-in-out.

```
<head> <style>
```

```
div {width: 100px;height: 100px; margin: 40px 30px; border: 1px solid #333;
```

```
background-color: #ccc; transition-property: background-color, width;
```

```
transition-duration: 2s, 10s;transition-timing-function: ease-in-out; }  
div:hover{ background-color: red; width: 200px;}  
</style> </head><body> <div></div></body>
```

Реализация данного кода представлена на рисунках 7.10-7.11.

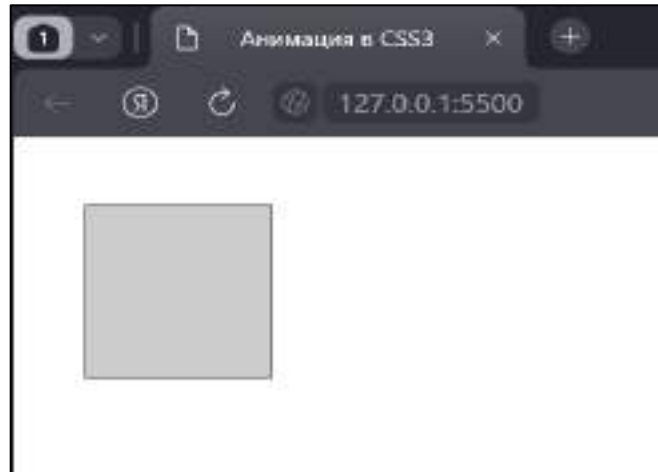


Рисунок 7.10 - Переход до

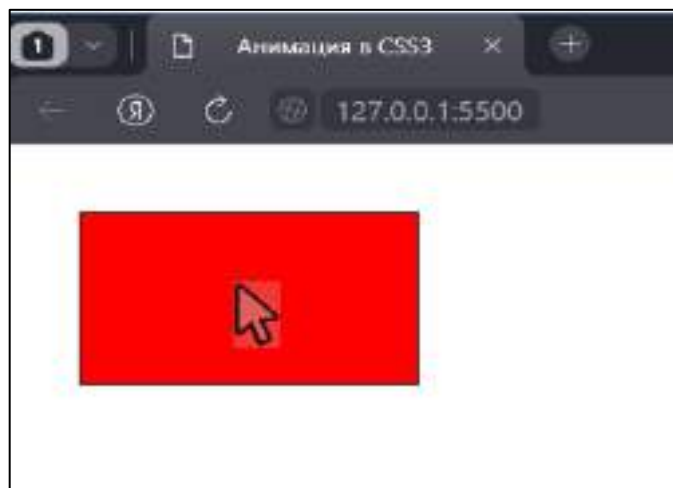


Рисунок 7.11 - Переход после

7.2.3 Задержка перехода

Свойство `transition-delay` позволяет определить задержку перед выполнением перехода:

```
transition-delay: 500ms;
```

Временной период также указывается в секундах (s) или миллисекундах (ms).

7.2.4 Свойство transition

Свойство transition представляет сокращенную запись выше рассмотренных свойств. Например, следующее описание свойств:

transition-property: background-color;

transition-duration: 3s;

transition-timing-function: ease-in-out;

transition-delay: 500ms;

Будет аналогично следующей записи:

transition: background-color 3s ease-in-out 500ms;

7.3 Анимация в CSS3

Анимация дает больше возможности изменить стиль элемента. При переходе есть набор свойств с начальными значениями, которые имеет элемент до начала перехода, и конечными значениями, которые устанавливаются после завершения перехода. Однако при использовании анимации можно иметь не только два набора значений - начальный и конечный, но и множество промежуточных наборов значений.

Анимация предполагает последовательную смену ключевых кадров. Каждый ключевой кадр определяет один набор значений свойств для анимации. И последовательная смена таких кадров фактически будет представлять собой анимацию.

Объявление ключевого кадра в CSS3 имеет следующую форму:

```
@keyframes название_анимации {  
  from {/* начальные значения свойств CSS */ }  
  to {/* конечные значения свойств CSS */}}
```

Название анимации отображается после ключевого слова @keyframes. Затем определяются как минимум два ключевых кадра в фигурных скобках. Блок после

ключевого слова `from` объявляет начальный ключевой кадр, а после ключевого слова `to` блок определяет конечный ключевой кадр. В каждом ключевом кадре определяется одно или несколько свойств CSS, как и при создании обычного стиля.

Далее приведен пример анимации для смены фонового цвета элемента.

```
<head> <style>
    @keyframes backgroundColorAnimation {
        from {background-color: blue; } to { background-color: purple; } }
    div{ width: 100px; height: 100px; margin: 40px 30px;
        border: 1px solid #333; background-color: #ccc;
        animation-name: backgroundColorAnimation;animation-duration: 2s;}
</style> </head> <body> <div>
</div> </body>
```

Реализация данного кода представлена на рисунках 7.12-7.13.

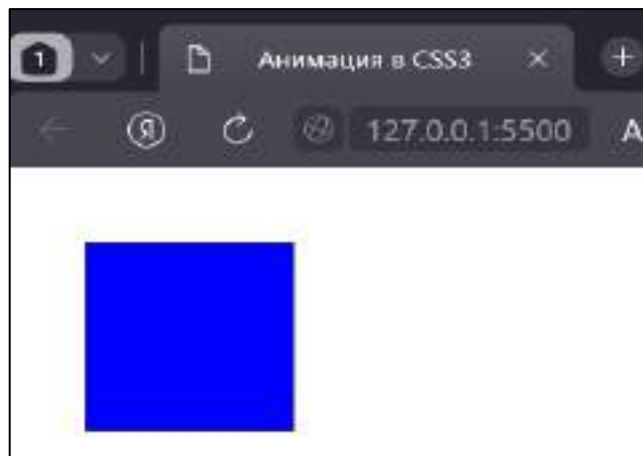


Рисунок 7.12 - Анимация до

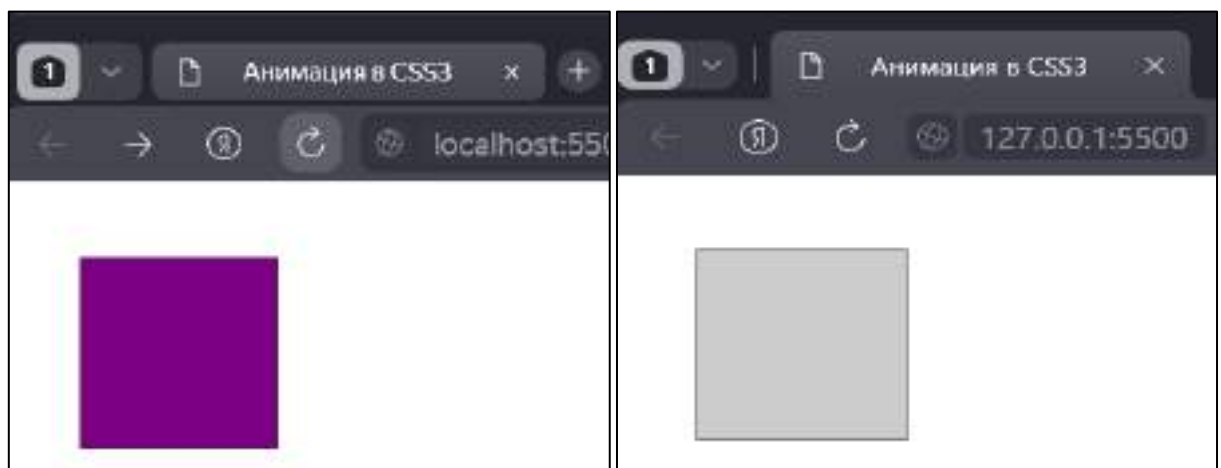


Рисунок 7.13 - Анимация после

В данном случае анимация называется `backgroundColorAnimation`. Эта анимация обеспечивает переход от синего цвета фона к фиолетовому цвету фона. После завершения анимации будет установлен цвет, определенный для элемента `div`.

Чтобы прикрепить анимацию к элементу, используется свойство `animation-name` в его стиле. Значением этого свойства является имя применяемой анимации.

Также с помощью свойства `animation-duration` необходимо задать время анимации в секундах или миллисекундах.

При таком определении анимация начнется сразу после загрузки страницы. Однако также можно запускать анимацию на основе действий пользователя. Например, определив стиль псевдокласса `:hover`, можно указать, что анимация начнется при наведении указателя мыши на элемент.

```
@keyframes backgroundColorAnimation {  
  from {background-color: red; }  
  to { background-color: blue; }  
}  
div{ width: 100px; height: 100px;margin: 40px 30px;  
  border: 1px solid #333; background-color: #ccc;}  
div:hover{animation-name: backgroundColorAnimation; animation-duration: 2s;}
```

7.3.1 Множество ключевых кадров

Как уже выше говорилось выше, анимация кроме двух стандартных ключевых кадров позволяет задействовать множество промежуточных. Для определения промежуточного кадра применяется процентное значение анимации, в котором этот кадр должен использоваться:

```
@keyframes backgroundColorAnimation {  
  from {background-color: red; }  
  25%{ background-color: yellow;}  
  50%{ background-color: green;}  
  75%{background-color: blue;}  
  to {background-color: violet;}  
}
```

В данном случае анимация начинается с красного цвета. Через 25% времени анимации цвет меняется на желтый, еще через 25% - на зеленый и так далее (рисунки 7.14-7.16).

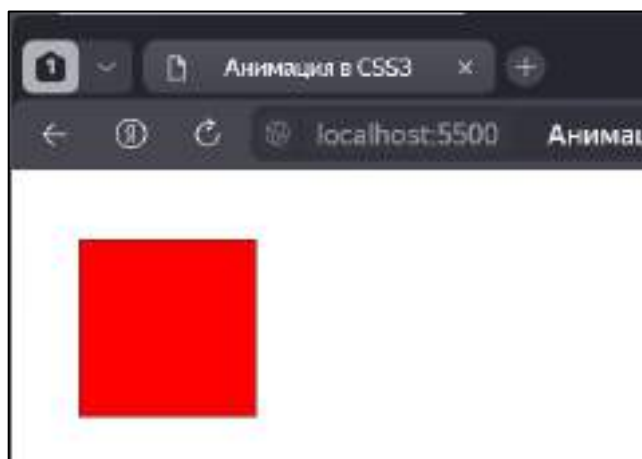


Рисунок 7.14 - До начала анимации

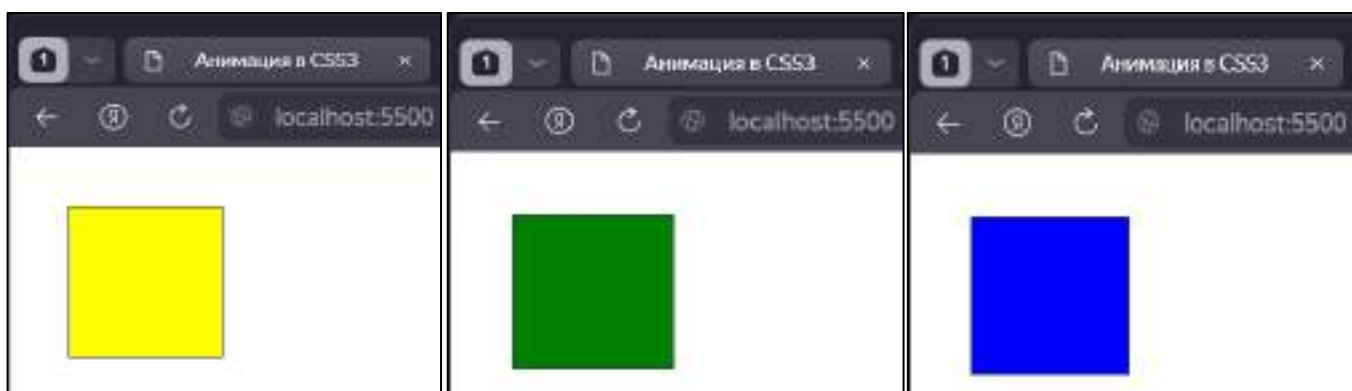


Рисунок 7.15 - Середина анимации

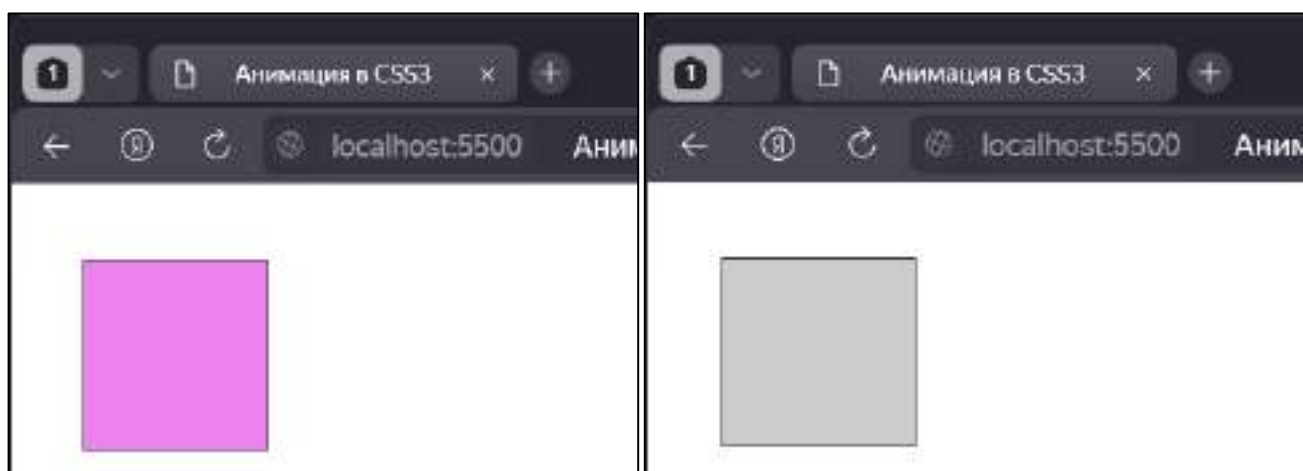


Рисунок 7.16 - Конец анимации

Также можно в одном ключевом кадре анимировать сразу несколько свойств.

```
@keyframes backgroundColorAnimation {  
  from { background-color: red; opacity: 0.2; }  
  to   { background-color: blue; opacity: 0.9; } }
```

Также можно определить несколько анимаций, но применять их вместе:

```
@keyframes backgroundColorAnimation {  
  from { background-color: red; }  
  to   { background-color: blue; }  
}  
  
@keyframes opacityAnimation {  
  from { opacity: 0.2; }  
  to   { opacity: 0.9; }  
}  
  
div{ width: 100px;  
  height: 100px;  
  margin: 40px 30px;  
  border: 1px solid #333;  
  background-color: #ccc;  
  animation-name: backgroundColorAnimation, opacityAnimation;  
  animation-duration: 2s, 3s; }
```

В качестве значения свойства `animation-name` через запятую перечисляются анимации, и также через запятую у свойства `animation-duration` задается время этих анимаций. Название анимации и ее время сопоставляются по позиции, то есть анимация `opacityAnimation` будет длиться 3 секунды.

7.3.2 Завершение анимации

В общем случае после завершения временного интервала, указанного у свойства `animation-duration`, завершается и выполнение анимации. Однако с помощью дополнительных свойств можно переопределить это поведение. Так, свойство `animation-iteration-count` определяет, сколько раз будет повторяться анимация. Например, три повтора анимации подряд можно задать следующим образом:

animation-iteration-count: 3;

Если необходимо, чтобы анимация запускалась бесконечное количество раз, то этому свойству присваивается значение `infinite`:

animation-iteration-count: infinite;

При повторе анимация будет начинаться снова с начального ключевого кадра.

С помощью свойства `animation-delay` можно определить задержку анимации:

animation-name: backgroundColorAnimation;

animation-duration: 5s;

animation-delay: 1s; /* задержка в 1 секунду */

7.3.3 Функция плавности анимации

Как и к переходам, к анимации можно применять те же функции плавности:

- `linear`: функция линейного сглаживания, изменяется равномерно с течением времени;

- `ease`: функция сглаживания, при которой анимация ускоряется к середине и замедляется к концу, обеспечивая более естественные изменения;

- `ease-in`: функция сглаживания, при которой вначале происходит только ускорение;

- `ease-out`: функция сглаживания, при которой происходит только ускорение в начале;

- `ease-in-out`: функция сглаживания, при которой анимация ускоряется к середине и замедляется к концу, обеспечивая более естественное изменение.

Для установки функции плавности применяется `animation-timing-function`:

```
@keyframes backgroundColorAnimation {  
  from {background-color: red; }  
  to { background-color: blue; }}  
div{ width: 100px;height: 100px; margin: 40px 30px; border: 1px solid #333;  
  animation-name: backgroundColorAnimation; animation-duration: 3s;  
  animation-timing-function: ease-in-out;}
```

7.3.4 Свойство animation

Свойство animation является сокращенным способом определения выше рассмотренных свойств:

animation: animation-name animation-duration animation-timing-function animation-iteration-count animation-direction animation-delay animation-fill-mode

Первые два параметра, которые предоставляют название и время анимации, являются обязательными. Остальные значения не обязательны.

Возьмем следующий набор свойств:

animation-name: backgroundColorAnimation;

animation-duration: 5s;

animation-timing-function: ease-in-out;

animation-iteration-count: 3;

animation-direction: alternate;

animation-delay: 1s;

animation-fill-mode: forwards;

Этот набор будет эквивалентен следующему определению анимации:

animation: backgroundColorAnimation 5s ease-in-out 3 alternate 1s forwards;

7.3.5 Создание баннера с анимацией

В качестве примера использования анимации описывается создание баннера.

```
<head><style> @keyframes text1
    {10%{opacity: 1;}
      40%{opacity: 0;}}
    @keyframes text2
    {30%{opacity: 0;}
      60%{opacity:1;}}
    @keyframes banner
    {10%{background-color: #008000;}
      40%{background-color: #FFFF00;}
      80%{background-color: #B22222;}}
```

```

.banner {width: 600px; height: 120px;
  background-color: #B22222; margin: 0 auto; position: relative;}
.text1,.text2 { position: absolute;
  width: 100%; height: 100%; line-height: 120px;
  text-align: center;font-size: 40px; color: white; opacity: 0;}
.text1 { animation : text1 6s infinite;}
.text2 { animation : text2 6s infinite;}
.animated { opacity: 0.8; position: absolute; width: 100%;height: 100%;
  background-color: #B22222; animation: banner 6s infinite;} </style></head><body>
<div class="banner"> <div class="animated">
  <div class="text1">Внимание</div>
  <div class="text2">Важная информация</div> </div> </div></body>

```

Реализация данного кода представлена на рисунках 7.17-7.18.

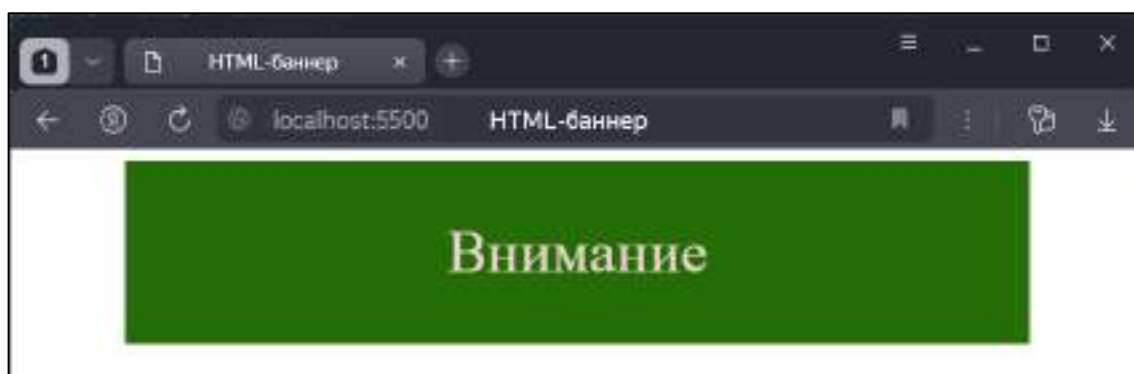


Рисунок 7.17 - Баннер до внедрения анимации

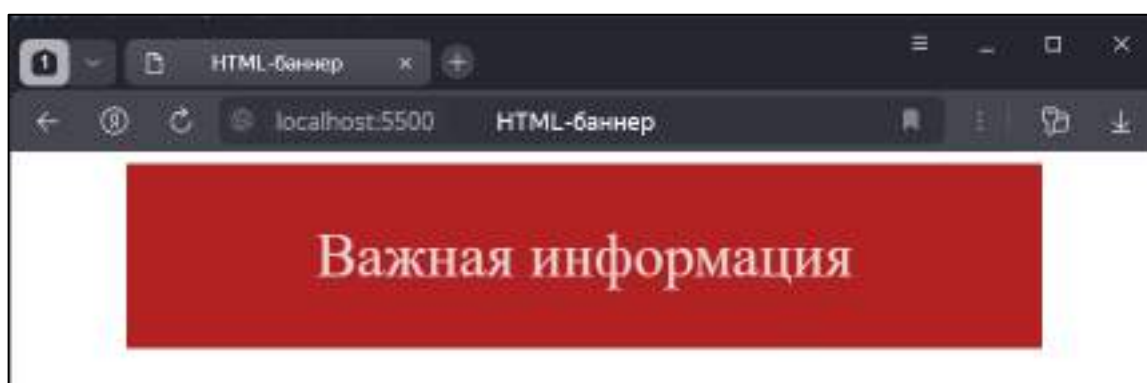


Рисунок 7.18- Баннер после внедрения анимации

В данном примере одновременно срабатывают три анимации. Анимация “banner” изменяет цвет фона баннера, а анимации text1 и text2 отображают и скрывают

текст с помощью настроек прозрачности. Когда первый текст виден, второй не виден и наоборот. Тем самым получается анимация текста в баннере.

7.4 Контрольные вопросы

- 1) Трансформации. Вращение. Масштабирование. Перемещение. Наклон.
- 2) Переходы. Переходы нескольких свойств.
- 3) Переходы. Функции переходов.
- 4) Анимация в CSS3. Множество ключевых кадров
- 5) Анимация в CSS3. Функция плавности анимации

7.5 Тестирование по главе

- 1) Для создания трансформаций применяется свойство ...
 - а) transformed;
 - б) transform;
 - в) transition;
 - г) visibility.

- 2) Для масштабирования применяется свойство ...
 - а) :rotate;
 - б) :scale;
 - в) :translate;
 - г) :skew.

- 3) Для наклона объекта применяется свойство ...

- а) :rotate;
- б) :scale;
- в) :translate;
- г) :skew.

4) Для перемещения применяется свойство ...

- а) :rotate;
- б) :scale;
- в) :translate;
- г) :skew.

5) Анимация от одного стиля к другому в течение определенного периода времени - это ...

- а) трансформация;
- б) переход;
- в) анимация;
- г) периодизация.

6) Чтобы указать свойство как анимируемое, его название передается свойству

- а) transform-property;
- б) transition-property;
- в) animation-property;
- г) transition-duration.

7) ... - функция плавности, при которой анимация ускоряется к середине и замедляется к концу

- а) ease-in-out;
- б) ease-in;
- в) ease-out;
- г) ease.

8) Какое свойство определяет задержку перед выполнением перехода?

- a) transition-property;
- б) transition-duration;
- в) transition-timing-function;
- г) transition-delay.

9) Какое свойство задает длительность анимации?

- a) animation-property;
- б) animation-duration;
- в) animation-iteration-count;
- г) animation-direction.

10) Какое свойство определяет, сколько раз будет повторяться анимация?

- a) animation-delay;
- б) animation-duration;
- в) animation-iteration-count;
- г) animation-direction.

8 Адаптивная верстка

8.1 Введение в адаптивный дизайн

В наше время все большее распространение получают различные гаджеты: смартфоны, планшеты, умные часы и другие устройства, позволяющие выходить в Интернет и просматривать содержимое веб-сайтов. Такое распространение гаджетов открывает новые возможности для разработки веб-сайтов, привлечения новых посетителей, продвижения информационных услуг и т. д. Но в то же время появляется новая проблема. Основная проблема заключается в том, что стандартная веб-страница будет выглядеть по-разному на разных устройствах с разным разрешением экрана. Первоначальным решением этой проблемы было создание специальных версий для мобильных устройств с использованием Media Query.

8.2 Media Query в CSS

Важным элементом в построении адаптивного дизайна являются правила Media Query, которые позволяют определить стиль в зависимости от размеров браузера пользователя [18].

Например, чтобы применить стиль только к мобильным устройствам можно написать так:

```
<html> <head><title>Адаптивная веб-страница</title>  
<meta name="viewport" content="width=device-width">  
<link rel="stylesheet" type="text/css" href="desctop.css" />  
<link      rel="stylesheet"      type="text/css"      media="(max-device-width:480px)"  
href="mobile.css" /></head> <body>
```

Значение атрибута `media (max-device-width:480px)` говорит о том, что стили из файла `mobile.css` будут применяться к тем устройствам, максимальная ширина экрана которых составляет 480 пикселей - то есть фактически это и есть мобильные устройства.

С помощью ключевого слова `and` можно комбинировать условия, например:

```
<link rel="stylesheet" type="text/css" media="(min-width:481px) and (max-width:768px)" href="mobile.css" />
```

Данный стиль будет применяться, если ширина браузера находится в диапазоне от 481 до 768 пикселей.

С помощью директивный `@import` можно определить один `css`-файл и импортировать в него стили для определенных устройств:

```
@import url(desktop.css);
```

```
@import url(tablet.css) (min-device-width:481px) and (max-device-width:768);
```

```
@import url(mobile.css) (max-device-width:480px);
```

Также можно не разделять стили по файлам, а использовать правила `CSS3 Media Query` в одном файле `css`:

```
body { background-color: red;}
```

```
/*Далее остальные стили*/
```

```
@media (max-device-width:480px){
```

```
  body {background-color: blue; }/*Далее остальные стили*/}
```

Применяемые функции в `CSS3 Media Query`:

- `aspect-ratio`: отношение ширины к высоте области отображения (браузера);
- `device-aspect-ratio`: отношение ширины к высоте экрана устройства;
- `max-width/min-width` и `max-height/min-height`: максимальная и минимальная ширина и высота области отображения (браузера);
- `max-device-width/min-device-width` и `max-device-height/min-device-height`: максимальная и минимальная ширина и высота экрана мобильного устройства;
- `orientation`: ориентация (портретная или альбомная).

Например, можно задать разные стили для разных ориентаций мобильных устройств:

/*Стили для портретной ориентации*/

@media only screen and (orientation: portrait){}

/*Стили альбомной ориентации*/

@media only screen and (orientation: landscape){}

Таким образом, меняется лишь определение стилей в зависимости от устройства, а сами стили CSS по сути остаются теми же, что используются для создания обычных сайтов.

Как правило, при определении стилей предпочтение отдается стилям для самых малых экранов - так называемый подход Mobile First, хотя это необязательно. Далее представлен пример создания адаптивной страницы.

```
<head>
  <style>
body { background-color: red; }
    @media (min-width: 481px) and (max-width:768px) {
      body { background-color: green;}}
    @media (min-width:769px) {
      body { background-color: blue;}}
  </style>
</head>
  <body>
    <h2>Адаптивная веб-страница</h2>
  </body>
```

Реализация данного кода представлена на рисунке 8.1.

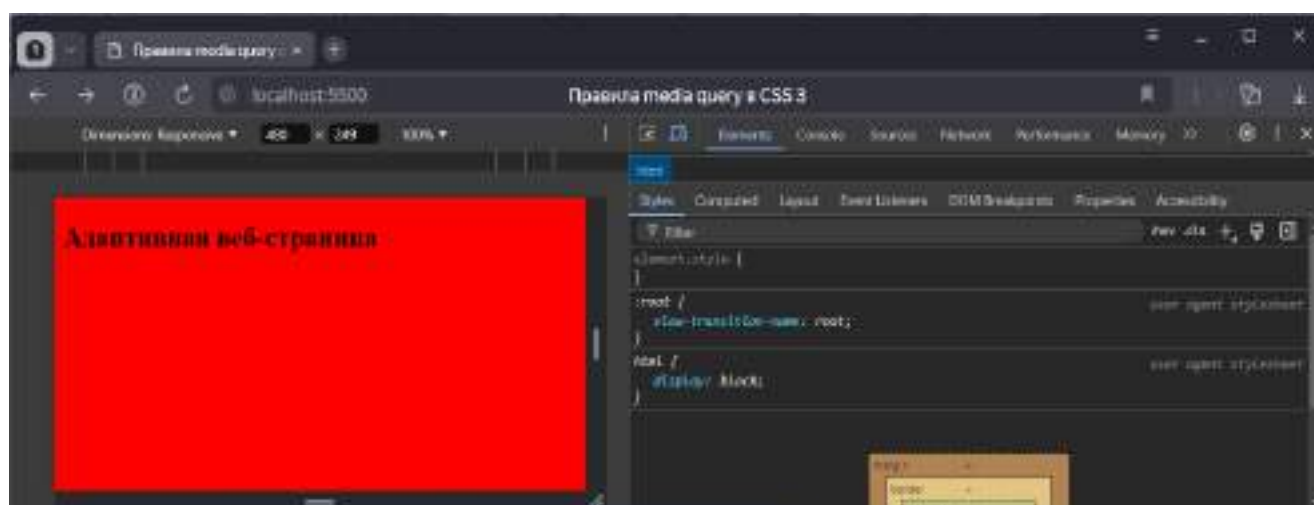


Рисунок 8.1 - Правила media query в CSS 3 для мобильного устройства

В данном примере сначала идут общие стили, которые актуальны прежде всего для мобильных устройств с небольшими экранами. Затем идут стили для устройств с экранами средней ширины: фаблеты, планшеты. И далее идут стили для десктопов.

И, например, на эмуляторе Opera Mobile при эмуляции устройства с шириной в 480 пикселей страница приобретет красный цвет.

А в браузере обычного компьютера страница будет иметь синий цвет, как и определено в стилях (рисунок 8.2).

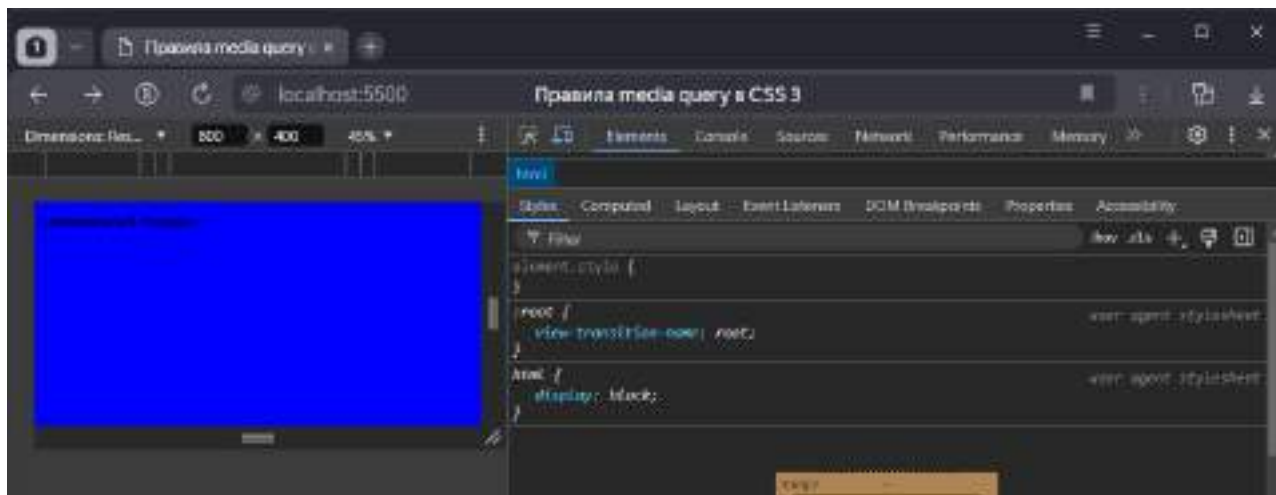


Рисунок 8.2 - Правила media query в HTML 5 для компьютера

В данном примере для задано одно свойство - цвет, который меняется, в зависимости от размеров устройства, либо компьютер, либо мобильное устройство, либо планшет (рисунок 8.3).

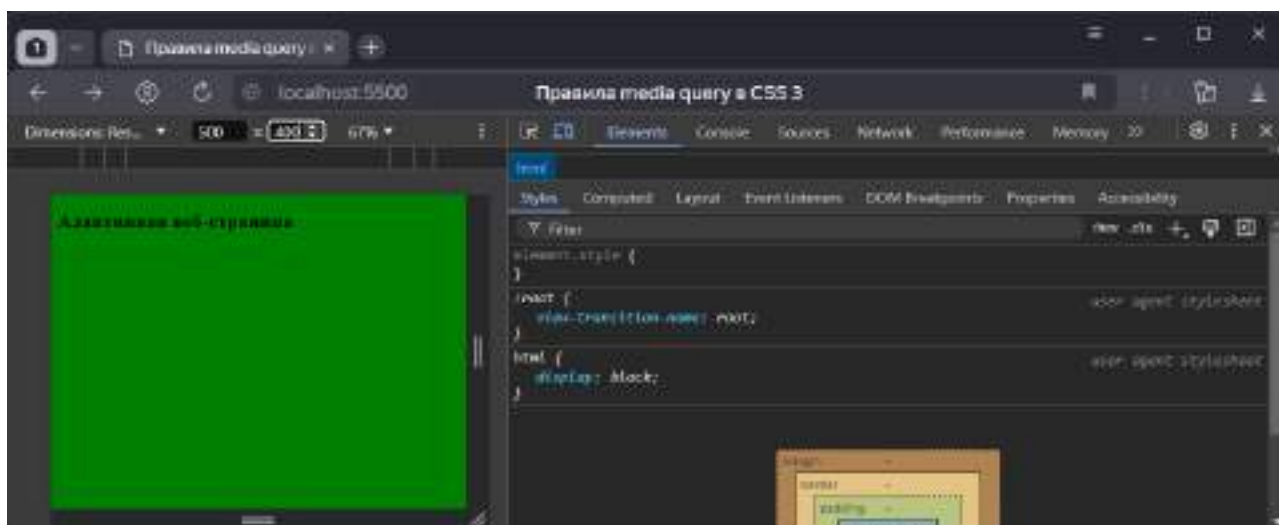


Рисунок 8.3 - Правила media query в HTML 5 для планшета

В нижеследующем примере представлено веб-приложение, в котором изменены свойства всех тегов. Помимо этого, адаптация происходит для трех видов устройств: компьютера, планшета и смартфона. Код CSS файла представлен далее.

Mobile.css

```
/* Стили для мобильных устройств (ширина экрана до 768px) */
@media screen and (max-width: 768px) {
  /* Здесь добавляются стили для мобильных устройств */
  /* Например, можно изменить размер шрифта, расположение элементов и другие стили */
  /* Примеры: */
  /* Изменение размера текста */
  body { font-size: 16px;} /* Изменение отступов и расположения */
  .container { padding: 10px; margin: 0;}
  /* Скрытие некоторых элементов, если необходимо */
  .hidden-on-mobile { display: none;}
  /* Уменьшение размеров изображений */
  .cookie-image { max-width: 100%;
  /* Максимальная ширина изображения 100% от родительского контейнера */
  height: auto;
  /* Автоматический расчет высоты, чтобы сохранить соотношение сторон */ }

  /* Стили для навбара на мобильных устройствах */
  nav { display: block; text-align: center;}
  nav h1 {font-size: 24px;
  /* Уменьшаем размер заголовка */
  margin-bottom: 20px;
  /* Добавляем отступ снизу */}
  nav a { display: block;margin-bottom: 10px; /* Добавление отступа между ссылками */}
  table { width: 100%; /* Ширина таблицы на всю ширину экрана */
  border-collapse: collapse;
  /* Убираем границы между ячейками */}
  th, td { padding: 10px; /* Добавляем отступы внутри ячеек */
  text-align: left;
  /* Выравниваем текст в ячейках слева */}
  th { background-color: #f2f2f2; /* Цвет фона для заголовков */ }
  tr:nth-child(even) {background-color: #f2f2f2;
  /* Чередуем цвет фона строк */ } }
```

```

/* Стили для анкеты на мобильных устройствах */
/* Выравнивание заголовка по центру */
h2 {text-align: center; }
/* Убираем отступ снизу у элементов формы */
label, input, textarea, div, legend { margin-bottom: 10px; }
/* Увеличиваем размер шрифта для полей ввода */
input[type="text"], textarea {font-size: 16px; }
/* Уменьшаем размер шрифта для меток и радиокнопок */
label,input[type="radio"] { font-size: 16px; }
/* Убираем лишние отступы у радиокнопок и чекбоксов */
#cookie-form input[type="radio"],input[type="checkbox"]
{ margin-right: 5px; vertical-align: middle;
/* Увеличение размера чекбоксов в форме анкеты */
width: 70px; height: 70px;}
#cookie-form button { width: 100%; }
/* Увеличиваем высоту текстового поля */
textarea { height: 100px;}
/* Уменьшаем отступ у кнопки отправки */
button { margin-top: 10px;}
/* Уменьшаем отступ у элементов внутри легенды */
legend div { margin-bottom: 5px;}}
/* Стили для планшетов (средних экранов) */
@media screen and (min-width: 768px) and (max-width: 1024px) {
/* Пример изменения размера текста и отступов для планшетов */
h2 { font-size: 24px; margin-top: 20px;}
/* Пример изменения размера чекбоксов для планшетов */
input[type="checkbox"] {
/* Пример изменения размера текстового поля для планшетов */
textarea { height: 150px;}
/* Пример изменения размера кнопки для планшетов */
button {font-size: 18px;}
table {width: 80%;
/* Уменьшаем ширину таблицы */
font-size: 14px; /* Уменьшаем размер шрифта текста в таблице */}
th, td {padding: 5px; /* Уменьшаем отступы внутри ячеек */}}

```

Реализация данного кода представлена на рисунках 8.4-8.6.

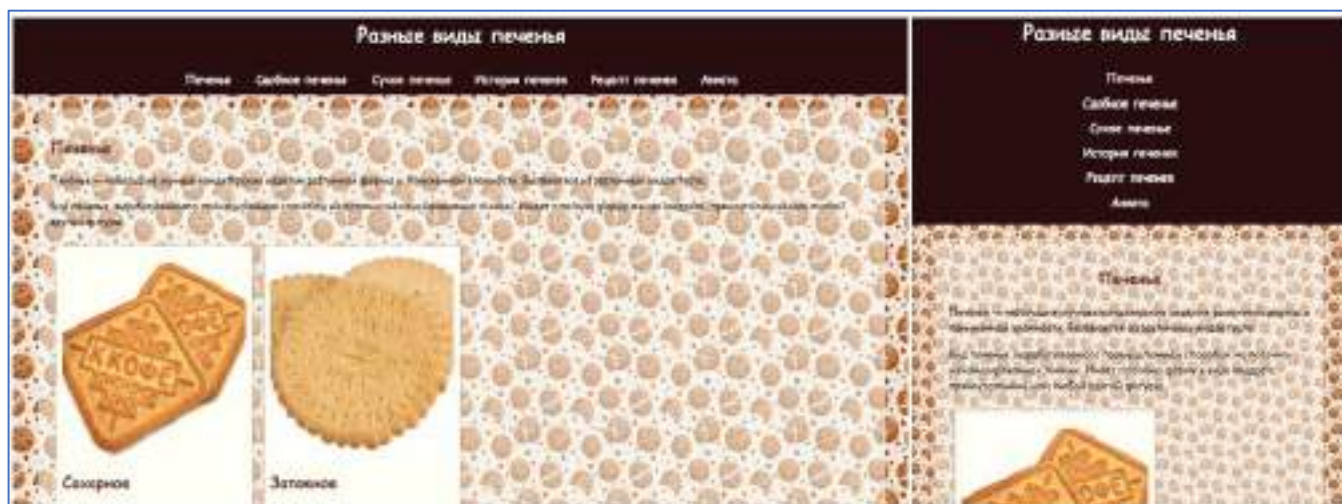


Рисунок 8.4 – Адаптация бургер-меню

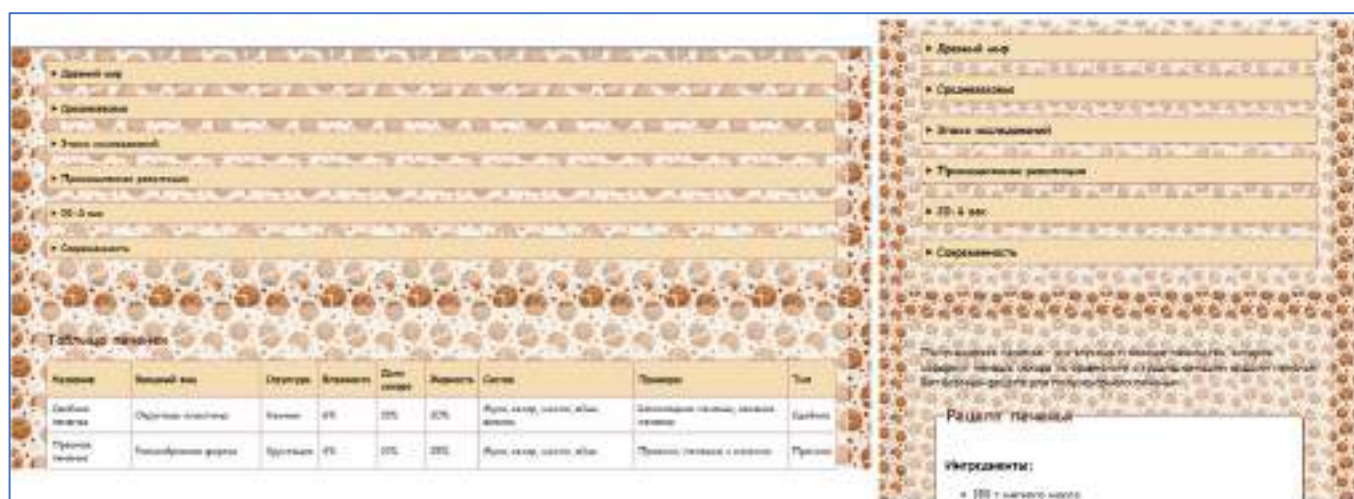


Рисунок 8.5 – Адаптация таблиц



Рисунок 8.6 – Адаптация размеров элементов checkbox

8.3 Контрольные вопросы

- 1) Основные понятия адаптивного дизайна.
- 2) Задание стилей Media Query в CSS.
- 3) Привести примеры задания стилей Media Query для мобильных телефонов.
- 4) Привести примеры задания стилей Media Query для планшетов.
- 5) Привести примеры задания стилей Media Query для компьютеров.

8.4 Тестирование по главе

- 1) Концепция адаптивного дизайна возникла на основе ...
 - а) необходимости подстраивать веб-страницы для различных устройств;
 - б) развития языка html;
 - в) развития CSS;
 - г) развития различных технологий.
- 2) Как происходит тестирование адаптивных веб-страниц?
 - а) тестирование на различных устройства;
 - б) тестирование при помощи эмулятора;
 - в) изменением размеров в коде;
 - г) тестирование не происходит.
- 3) Правила ... позволяют определить стиль в зависимости от размеров браузера пользователя
 - а) адаптивного дизайна;
 - б) Media Query;
 - в) кей-фреймов;

г) media.

4) Правило ... указывает, что стили применяются к мобильным устройствам

а) media="handheld";

б) media="screen";

в) media="all";

г) media="print".

5) Правило ... указывает, что стили будут применяться ко всем устройствам

д) media="handheld";

е) media="screen";

ж) media="all";

з) media="print".

6) При помощи директивы ... можно определить css-файл и поместить в него стили для определенных устройств

а) @export;

б) @import;

в) #export;

г) #import.

7) Функция ... - отношение ширины к высоте области браузера.

а) aspect-ratio;

б) device-aspect-ratio;

в) orientation;

г) width.

8) Функция ... - отношение ширины к высоте экрана устройства.

а) aspect-ratio;

б) device-aspect-ratio;

в) orientation;

г) width.

9) Функция ориентации ...

а) aspect-ratio;

б) device-aspect-ratio;

в) orientation;

г) width.

10) Функция ... - определение ширины.

а) aspect-ratio;

б) device-aspect-ratio;

в) orientation;

г) width.

9 Создание гибкого макета страницы с помощью Flexbox

9.1 Понятие Flexbox. Контейнер Flex

Flexbox - это общее название модуля Flexible Box Layout. Этот модуль определяет специальный режим верстки пользовательского интерфейса, который называется Flex Layout, отличающийся от макета таблицы или блока.

Благодаря Flexbox стало проще создавать сложные интерфейсы, в которых можно переопределять направление и выравнивание элементов, а также создавать адаптивные табличные представления. Все современные браузеры, включая Microsoft Edge, Opera, Яндекс, Safari, Firefox имеют полную поддержку этого модуля.

Основными компонентами макета Flexbox являются контейнер Flex (flex container) и элементы Flex (flex items).

9.1.1 Основные понятия

Одним из ключевых понятий является main axis или центральная ось. Это условная ось в гибком контейнере, вдоль которой располагаются гибкие элементы (рисунок 9.1).

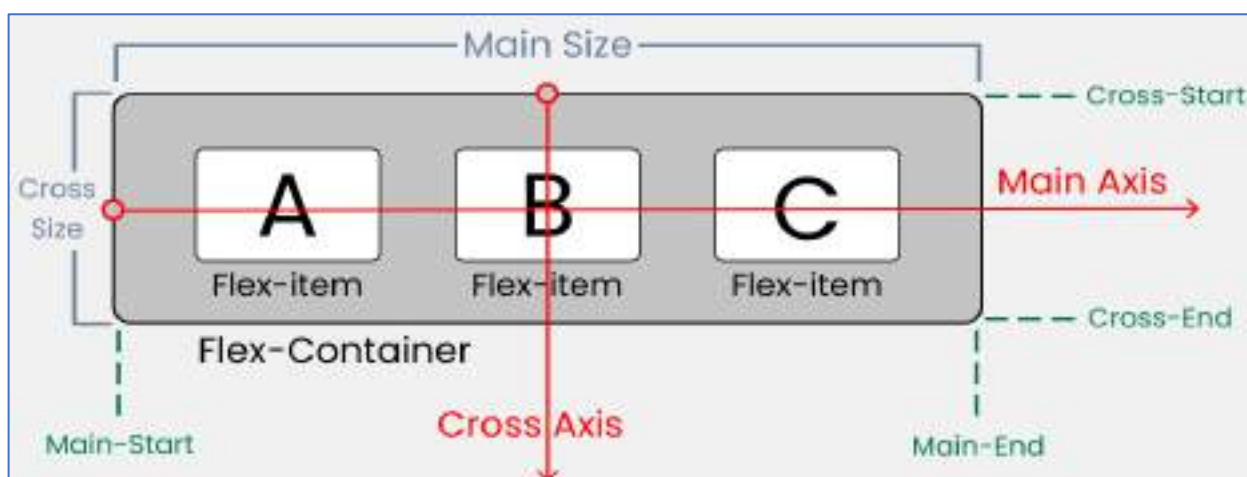


Рисунок 9.1 - Main axis и cross axis в flexbox и CSS 3

Элементы в контейнере можно располагать горизонтально в ряд и вертикально в столбец. В зависимости от типа локации будет меняться и центральная ось. Термины `main start` и `main end` описывают начало и конец центральной оси соответственно, а расстояние между ними называется основным размером (`main size`).

Помимо главной оси, существует еще `cross axis` или поперечная ось. Она перпендикулярна основной. При расположении элементов в ряд поперечная ось направлена сверху вниз, а при расположении в столбик - слева направо. Начало поперечной оси обозначалось как поперечное начало, а конец - как поперечный конец. Расстояние между ними называется `cross size`.

То есть, если элементы расположены в линию, основным размером представляет собой ширину контейнера или элементов, а поперечный размер представляет их высоту. С другой стороны, если элементы расположены в столбце, основным размером представляет собой высота контейнера и элементов, а поперечный размер представляет их ширину.

9.1.2 Создание flex-контейнера

Для создания flex-контейнера необходимо присвоить его стилевому свойству `display` одно из двух значений: `flex` или `inline-flex`.

Далее представлен пример создания простейшей веб-страницы, которая применяет flexbox. Для контейнера `flex-container` установлено свойство `display: flex`. В нем располагается три flex-элемента.

```
<head> <style>
.flex-container { display: flex; }
.flex-item {text-align:center; font-size: 1.2em; padding: 1.6em;color: snow; }
.color1 {background-color: RosyBrown;}
.color2 {background-color: SandyBrown ;}
.color3 {background-color: SaddleBrown;}
</style> </head>
<body><div class="flex-container">
<h3>Виды кофе</h3>
```

```
<div class="flex-item color1">Капучино. Капучино - кофейный напиток итальянской  
кухни на основе эспрессо с добавлением в него подогретого до 65 градусов  
вспененного молока. </div>
```

```
<div class="flex-item color2">Латте. Латте - кофейный напиток на основе молока,  
представляющий собой трёхслойную смесь из молочной пены, молока и кофе эспрессо. </div>
```

```
<div class="flex-item color3">Американо. Американо - кофе по-американски, способ  
приготовления кофе, заключающийся в соединении определённого количества  
горячей воды и эспрессо. </div> </div></body>
```

Реализация данного кода представлена на рисунке 9.2.

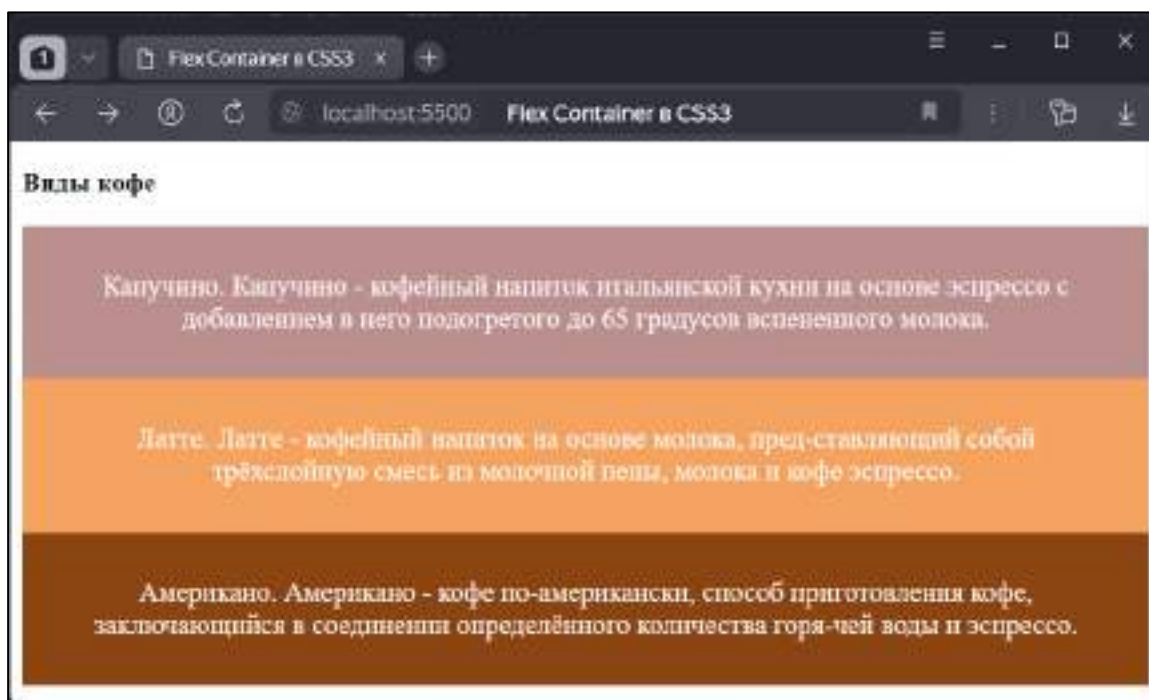


Рисунок 9.2 - Flex Container в CSS3

Если значение flex определяет контейнер как блочный элемент, то значение inline-flex определяет элемент как строчный (inline). Далее представлено оба способа на примере веб-страницы о чае.

```
<head> <style> .flex-container { display: flex; border:3px solid black; }  
.inline-flex-container { display: inline-flex; border:3px solid black; margin-top:8px; }  
.flex-item { text-align:center; font-size: 1.2em; padding: 1.6em; }  
.color1 {background-color: BurlyWood}  
.color2 {background-color: YellowGreen;}  
.color3 {background-color: AntiqueWhite;} </style></head>
```

```

<body> <div> <div class="flex-container">
  <div class="flex-item color1">Чёрный чай</div>
  <div class="flex-item color2">Зеленый чай</div>
  <div class="flex-item color3">Белый чай</div> </div>
<div class="inline-flex-container">
  <div class="flex-item color1">Чёрный чай</div>
  <div class="flex-item color2"> Зеленый чай</div>
  <div class="flex-item color3">Белый чай</div> </div> </div> </body>

```

Реализация данного кода представлена на рисунке 9.3.



Рисунок 9.3 - Inline-flex и flexbox в CSS3

В частности, в первом случае гибкий контейнер растягивается на всю ширину страницы, а во втором случае занимает ровно то пространство, которое необходимо для гибких элементов.

9.2 Направление flex-direction

Flex-элементы во flex-контейнере могут иметь определенное направление, а именно они могут располагаться в виде строк или в виде столбцов. Для управления направлением элементов CSS3 предоставляет свойство flex-direction. Оно определяет направление элементов и может принимать следующие значения:

- row: значение по умолчанию, при котором элементы располагаются в виде строки слева направо;
- row-reverse: элементы также располагаются в виде строки только в обратном порядке справа налево;
- column: элементы располагаются в столбик сверху вниз;
- column-reverse: элементы располагаются в столбик в обратном порядке.

Далее приведен пример использования свойства flex-direction.

```
<head> <style> .flex-container { display: flex;border:2px solid black;}
        .row {flex-direction: row;}
        .row-reverse {flex-direction: row-reverse; }
        .flex-item {text-align:center;font-size: 1.2em;padding: 1.6em; }
        .color1 {background-color: Plum;}
        .color2 {background-color: LightPink;}
        .color3 {background-color: LightGreen;}
    </style></head>
```

```
<body> <h3>Строка в прямом направлении </h3>
```

```
<div class="flex-container row">
```

```
<div class="flex-item color1">Ирис - род многолетних корневищных растений
семейства Ирисовые, или Касатиковые. Род насчитывает около 800 видов с богатейшим
разнообразием форм и оттенков, за что и получил своё название (от др. греч. радуга).</div>
```

```
<div class="flex-item color2">Пион - род травянистых многолетников и листопадных
кустарников. Единственный род семейства пионовые, пионы цветут в конце весны, ценятся
садоводами за пышную листву, эффектные цветы и декоративные плоды. </div>
```

```
<div class="flex-item color3">Ландыш - монотипный либо олиготипный род
однодольных растений семейства Спаржевые. Как правило, считается монотипным родом с
единственным видом Ландыш майский. </div>
```

```
</div> <h3>Реверсивная строка</h3> <div class="flex-container row-reverse">
```

```
<div class="flex-item color1">Ирис - род многолетних корневищных растений
семейства Ирисовые, или Касатиковые. Род насчитывает около 800 видов с богатейшим
разнообразием форм и оттенков, за что и получил своё название (от др. греч. радуга).</div>
```

```
<div class="flex-item color2">Пион - род травянистых многолетников и листопадных
кустарников. Единственный род семейства пионовые, пионы цветут в конце весны, ценятся
садоводами за пышную листву, эффектные цветы и декоративные плоды. </div>
```

`<div class="flex-item color3">Ландыш - монотипный либо олиготипный род однодольных растений семейства Спаржевые. Как правило, считается монотипным родом с единственным видом Ландыш майский. </div> </div> </body>`

Реализация данного кода представлена на рисунке 9.4.



Рисунок 9.4 - Flex-direction во flexbox и CSS3

Аналогично работает расположение в виде столбца.

`<head> <style>`

`.flex-container { display: flex; border: 2px solid black; }`

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

`.column-reverse { flex-direction: column-reverse; }`

`.flex-item { text-align: center; font-size: 0.9em; padding: 0.9em; color: black; }`

`.color1 { background-color: Gold; }`

`.color2 { background-color: Tomato; }`

`.color3 { background-color: Chartreuse; } </style> </head>`

`<body> <h3>Столбец в прямом направлении</h3>`

`<div class="flex-container column">`

`<div class="flex-item color1"> Банан - название съедобных плодов культивируемых растений рода Банан (Musa); обычно под таковыми понимают Musa acuminata. С ботанической точки зрения банан является ягодой, многосеменной и толстокожей. </div>`

`<div class="flex-item color2">Яблоко – сочный плод яблони, который употребляется в пищу в свежем и запеченном виде, служит сырьём в кулинарии и для приготовления напитков. </div>`

Наибольшее распространение получила яблоня домашняя, реже выращивают яблоню сливолистную.

Груша - род плодовых и декоративных деревьев и кустарников семейства розовые, а также их плод. Плоды груш отличаются характерной зернистой мякотью из-за наличия каменных клеток.

Реверсивный столбец

Банан - название съедобных плодов культивируемых растений рода Банан (*Musa*); обычно под таковыми понимают *Musa acuminata*. С ботанической точки зрения банан является ягодой, многосеменной и толстокожей.

Яблоко – сочный плод яблони, который употребляется в пищу в свежем и запеченном виде, служит сырьём в кулинарии и для приготовления напитков. Наибольшее распространение получила яблоня домашняя, реже выращивают яблоню сливолистную.

Груша - род плодовых и декоративных деревьев и кустарников семейства розовые, а также их плод. Плоды груш отличаются характерной зернистой мякотью из-за наличия каменных клеток.

Реализация данного кода представлена на рисунке 9.5.

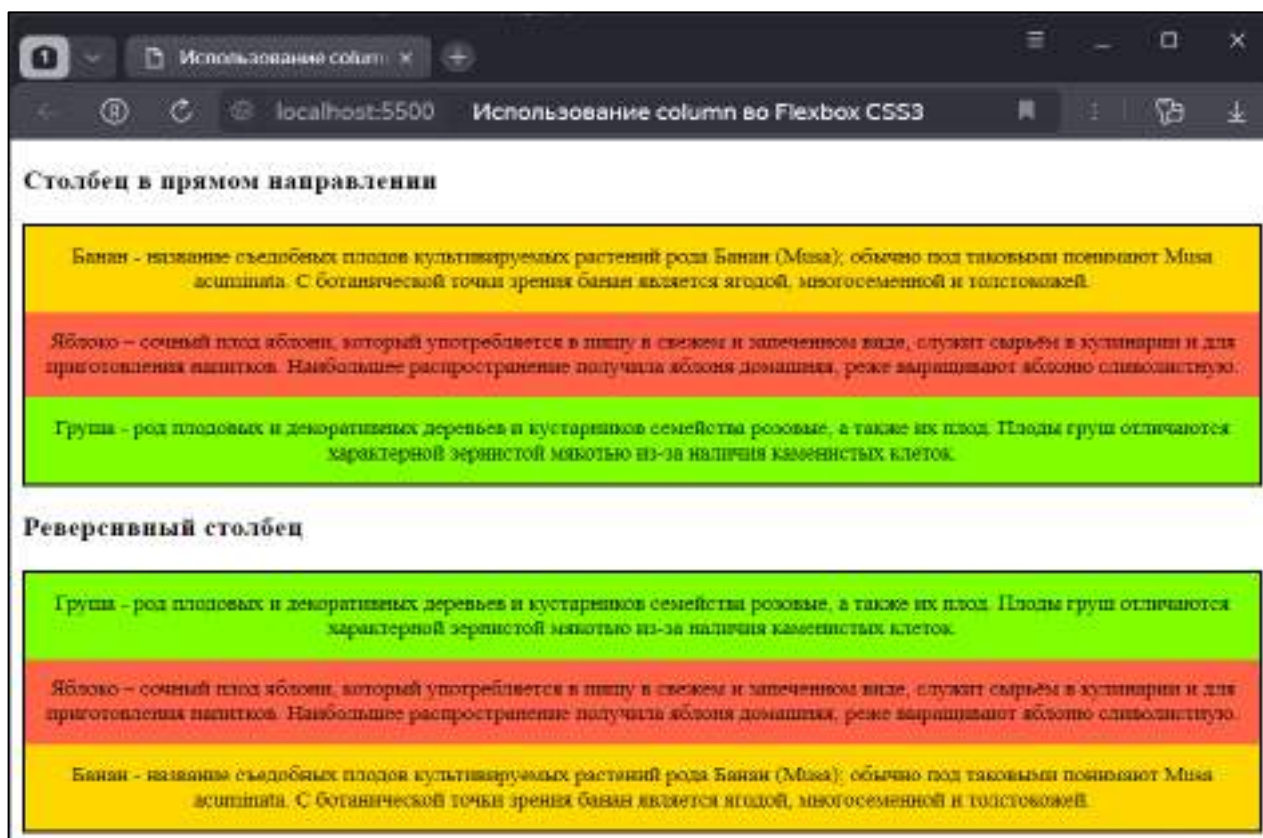


Рисунок 9.5 – Использование column во Flexbox CSS3

9.3 Flex-flow. Порядок элементов

9.3.1 Flex-flow

Свойство flex-flow является сокращенной формой записи свойств flex-direction и flex-wrap и позволяет за одну декларацию задать направление главной оси и возможность переноса флекс-элементов на новую строку.

Свойство flex-wrap определяет, должны ли элементы переноситься на новую строку или оставаться на одной. Имеет три значения - nowrap, wrap, wrap-reverse. nowrap - значение по умолчанию, указывает, что элементы располагаются в контейнере в одну строку или столбец.

```
<head> <style>    .flex-container {display: flex;
                    border: solid 0.24em black;
                    height:24.24em; flex-flow: row wrap;}
                    .flex-item {text-align:center;
                                font-size: 0.9em; padding: 1.4em; }
.col1 {background-color: PaleTurquoise;}
.col2 {background-color: LightSkyBlue;}
.col3 {background-color: LightBlue;}
.col4 {background-color: DeepSkyBlue;}
.col5 {background-color: CornflowerBlue;} </style></head>
<body><div class="flex-container">
    <div class="flex-item col1">Яхта - судно, оснащенное парусами или оборудованное
механическим двигателем, предназначенное для прогулок, туризма или спорта.</div>
    <div class="flex-item col2">Теплоход - обобщающее понятие, описывающее класс
самоходных судов, преобразующих тепловую энергию сжигания топлива в
механическую.</div>
    <div class="flex-item col3">Катер – открытая модель судна, с мотором стационарного типа
или с подвесным движком, обустроенные каютами.</div>
    <div class="flex-item col4"> Пароход – судно, движение которого обеспечивается паровой
машиной или турбиной вращающей гребные колеса или винты.</div>
```



```
<div class="flex-item col5"> Корабль – в современном понимании, военное или большое парусное судно с боевыми орудиями. </div> </div> </body>
```

Реализация данного кода представлена на рисунке 9.6.



Рисунок 9.6 – Применение свойства Flex-flow

9.3.2 Свойство order

Свойство `order` позволяет установить группу для flex-элемента, позволяя тем самым переопределить его позицию внутри flex-контейнера. В качестве значения свойство принимает числовой порядок группы. К одной группе может принадлежать несколько элементов.

Например, элементы в группе 0 располагаются перед элементами с группой 1, а элементы с группой 1 располагаются перед элементами с группой 2 и так далее.

```
<head> <style>.flex-container { display: flex; flex-flow: row wrap;}
    .flex-item {text-align:center; font-size: 1em; padding: 1.5em; }
    .gr1{order:-1; } .gr2{ order:1;}
    .col1 {background-color: Tan;}
    .col2 {background-color: BurlyWood;} .col3 {background-color: Khaki;}
    .col4 {background-color: PeachPuff;} .col5 {background-color: DarkKhaki ;}
</style></head> <body> <h3>Породы обезьян</h3>
```

```

<div class="flex-container">
  <div class="flex-item col1">Мартышка</div>
  <div class="flex-item col2 gr2">Шимпанзе</div>
  <div class="flex-item col3 gr2">Орангутан</div>
  <div class="flex-item col4">Макака</div>
  <div class="flex-item col5 gr1">Горилла</div> </div></body>

```

Реализация данного кода представлена на рисунке 9.7

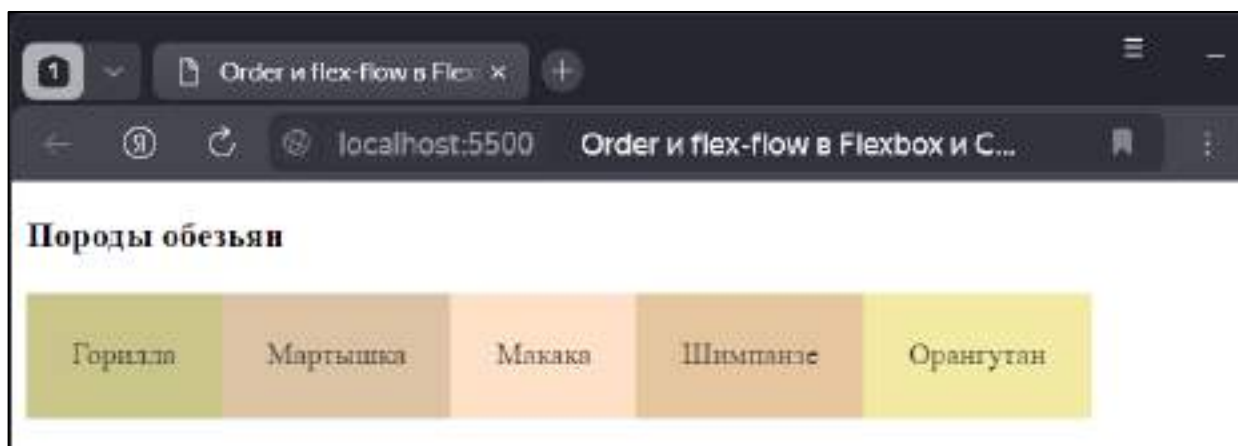


Рисунок 9.7 - Order и flex-flow в Flexbox и CSS3

В данном случае определены 3 группы. Первый отображается последний элемент, так как он имеет группу -1. По умолчанию если у элементов явным образом не указано свойство order, то оно имеет значение 0. И последними в данном случае отображаются второй и третий элемент, так как у них свойство order равно 1.

9.4 Выравнивание элементов. Justify-content

Иногда можно сталкиваться с ситуацией, когда пространство flex-контейнеров по размеру отличается от пространства, необходимого для flex-элементов. Например, когда flex-элементы не используют все пространство flex-контейнера; либо flex-элементам требуется большее пространство, чем доступно во flex-контейнере. В этом случае элементы выходят за пределы контейнера.

Для управления подобными ситуациями можно применять свойство `justify-content`. Оно выравнивает элементы вдоль основной оси - `main axis` и может принимать следующие значения:

- `flex-start`: значение по умолчанию, при котором первый элемент выравнивается по левому краю контейнера (при расположении в виде строки) или по верху (при расположении в виде столбца), за ним располагается второй элемент и так далее;

- `flex-end`: последний элемент выравнивается по правому краю (при расположении в виде строки) или по низу (при расположении в виде столбца) контейнера, за ним выравнивается предпоследний элемент и так далее;

- `center`: элементы выравниваются по центру;

- `space-between`: если в строке только один элемент или элементы выходят за границы `flex`-контейнера, то данное значение аналогично `flex-start`. В остальных случаях первый элемент выравнивается по левому краю (при расположении в виде строки) или по верху (при расположении в виде столбца), а последний элемент - по правому краю контейнера (при расположении в виде строки) или по низу (при расположении в виде столбца).

- `space-around`: если в строке только один элемент или элементы выходят за пределы контейнера, то его действие аналогично значению `center`. В ином случае элементы равным образом распределяют пространство между левым и правым краем контейнера, а расстояние между первым и последним элементом и границами контейнера составляет половину расстояния между элементами.

Далее приведен пример выравнивания элементов строки.

```
<head> <style>
```

```
  .flex-container { display: flex; border:2px black solid;}
```

```
  .end{justify-content: flex-end; }
```

```
  .center{ justify-content: center; }
```

```
  .between{ justify-content: space-between;}
```

```
  .around{ justify-content: space-around; }
```

```
  .flex-item {text-align:center;font-size: 0.9em; padding: 1.6em; }
```

```

.col1 {background-color: LightCoral;} .col4 {background-color: MediumTurquoise;}
.col2 {background-color: Orange ;} .col5 {background-color: Lime;}
.col3 {background-color: Cyan;} .col6 {background-color: DeepSkyBlue;}
</style> </head> <body> <h2> Материки планеты Земля </h2>
    <h3>Использование flex-end</h3>
<div class="flex-container end">
    <div class="flex-item col1">Азия</div>
    <div class="flex-item col2">Африка</div>
    <div class="flex-item col3">Европа </div>
    <div class="flex-item col4">Австралия </div>
    <div class="flex-item col5"> Америка </div>
    <div class="flex-item col6">Антарктида</div>
</div><h3>Использование center</h3>
<div class="flex-container center">
    <div class="flex-item col1">Азия </div>
    <div class="flex-item col2">Африка</div>
    <div class="flex-item col3">Европа</div>
    <div class="flex-item col4">Австралия</div>
    <div class="flex-item col5">Америка</div>
    <div class="flex-item col6">Антарктида</div>
<h3>Использование space-between</h3>
<div class="flex-container between">
    <div class="flex-item col1">Азия</div>
    <div class="flex-item col2">Африка</div>
    <div class="flex-item col3">Европа</div>
    <div class="flex-item col4">Австралия</div>
    <div class="flex-item col5">Америка</div>
    <div class="flex-item col6">Антарктида</div>
<h3>Использование space-around</h3>
<div class="flex-container around">
    <div class="flex-item col1">Азия</div>
    <div class="flex-item col2">Африка</div>
    <div class="flex-item col3">Европа</div>
    <div class="flex-item col4">Австралия</div>

```

```

<div class="flex-item col4">Америка</div>
<div class="flex-item col4">Антарктида</div> </div> </body>

```

Реализация программного кода представлена на рисунке 9.8.



Рисунок 9.8 - Justify-content в Flexbox и CSS3

Выравнивание в виде столбцов представлено на рисунке 9.9.



Рисунок 9.9 - Justify-content и column в Flexbox и CSS3

9.5 Выравнивание элементов. Свойство align-items

Свойство align-items также выравнивает элементы, но по поперечной оси (вертикально, если они организованы в виде строки, и горизонтально, если они организованы в виде столбца). Это свойство может принимать следующие значения:

- stretch: значение по умолчанию, при котором гибкие элементы растягиваются по всей высоте или по всей ширине гибкого контейнера;
- flex-start: элементы выравниваются по верхнему краю или по левому краю гибкого контейнера;
- flex-end: элементы выравниваются по нижнему краю или по правому краю гибкого контейнера;
- center: элементы выравниваются по центру гибкого контейнера;
- baseline: элементы выравниваются по их базовой линии.

Далее приведен пример выравнивания при расположении в строку.

```
<head> <style>
    .container { display: flex;border:2px black solid;height:4.9em;}
    .start{align-items: flex-start;}    .end{align-items: flex-end;}
    .center{align-items: center; }    .baseline{ align-items: baseline; }
    .item { text-align:center; font-size: 0.9em; padding: 1.1em; color: white; }
    .largest{ padding-top:1.9em; }
    .col1 {background-color: LimeGreen;} .col2 {background-color: MediumSeaGreen;}
    .col3 {background-color: ForestGreen;}. col4 {background-color: DarkOliveGreen;}
</style> </head>

<body> <h3>Использование flex-start</h3>
<div class="container start">
    <div class="item col1">Береза</div>
    <div class="item col2">Тополь</div>
    <div class="item col3">Дуб</div>
    <div class="item col4">Осина</div>
</div> <h3>Использование flex-end</h3> <div class="container end">
    <div class="item col1">Береза</div>
```

```

    <div class="item col2">Тополь</div>
    <div class="item col3">Дуб</div>
    <div class="item col4">Осина</div>
</div> <h3>Использование center</h3> <div class="container center">
    <div class="item col1">Береза</div>
    <div class="item col2">Тополь</div>
    <div class="item col3">Дуб</div>
    <div class="item col4">Осина</div>
</div> <h3>Использование baseline</h3><div class="container baseline">
    <div class="item col1">Береза</div>
    <div class="item col2 largest">Тополь</div>
    <div class="item col3">Дуб</div>
    <div class="item col4">Осина</div>
</div></body>

```

Реализация данного кода представлена на рисунке 9.10.

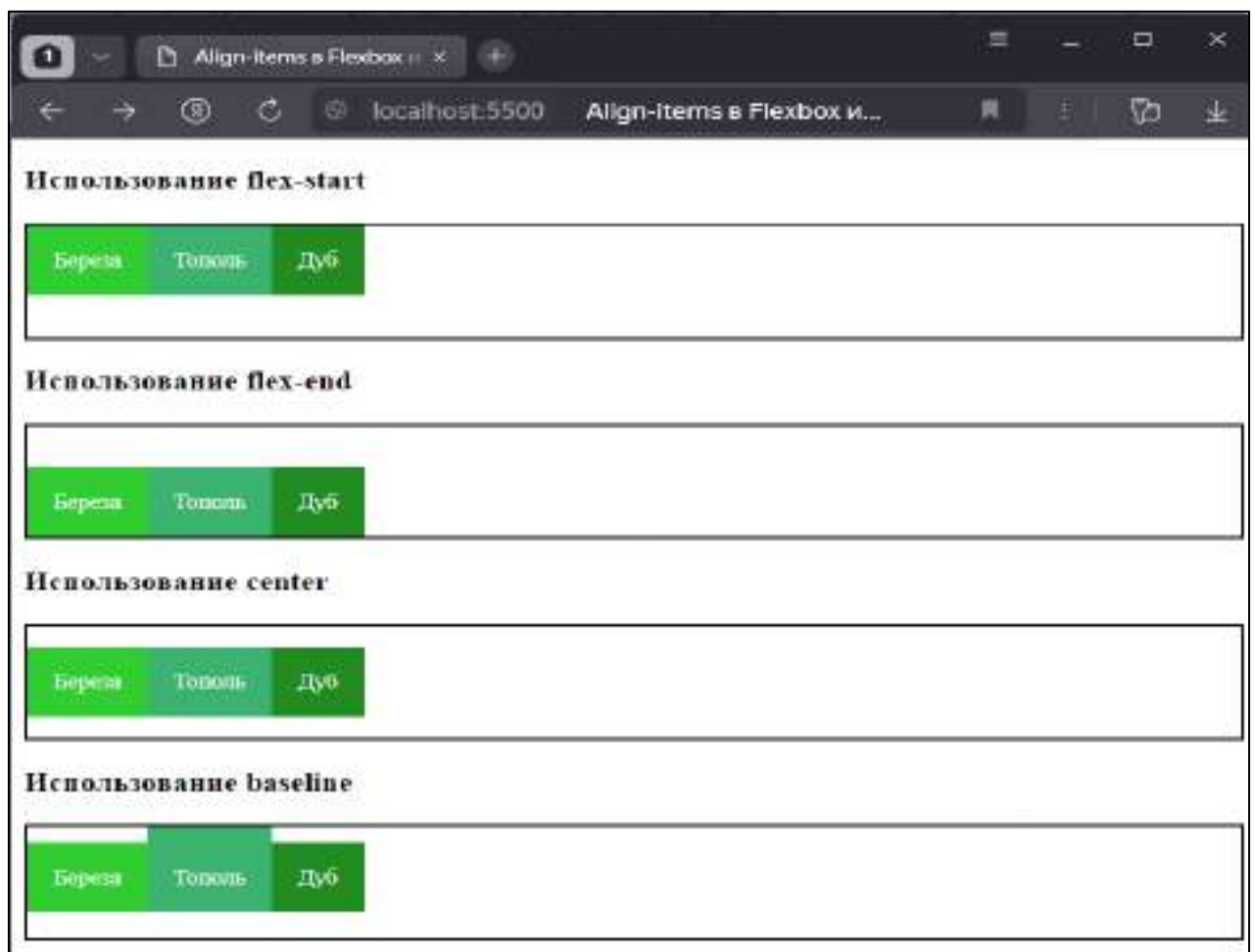


Рисунок 9.10 - Align-items во Flexbox и CSS3

Аналогично свойство работает при расположении в столбец. Например, если изменить стили flex-контейнера по столбцам, то получится результат, представленный на рисунке 9.11.



Рисунок 9.11 - Align-items и column в Flexbox и CSS3

9.6 Многоколоночный дизайн на Flexbox

Основная идея мультиколонок заключается в том, что можете взять фрагмент содержимого и поместить его в несколько колонок, как в газете. Вы делаете это с помощью одного или двух свойств. Далее рассматривается, как можно сделать простейшие многоколоночные макеты страницы с помощью Flexbox.

9.6.1 Двукколоночный дизайн

Далее представлен пример представления страницы, состоящей из двух колонок, заданных с помощью свойств flexbox.

```
<head> <style>
  *{ box-sizing: border-box; }
html, body {padding: 1; margin: 1; font-family: verdana, arial, sans-serif;}
```



```

body { display: flex; padding: 1.1em; flex-direction: column;}
.item {background-color: #9400D3; font-size: 1.1em; padding: 1em;}
.item:nth-child(even) {background-color: #DDA0DD; }
@media screen and (min-width: 600px) {body { flex-direction: row;}}
</style></head> <body><div class="item">
<h2>Певица Люся Чеботина</h2> <p>Родилась 26 апреля 1997 года в Петропавловске-
Камчатском. Окончила Государственный музыкальный колледж эстрадного и джазового
искусства. С 2016-го девушка записывала композиции собственного сочинения. 29 октября
2021 года выпустила второй студийный альбом “The End”. Ведущий сингл с альбома
“Солнце Монако” уже к середине ноября занял 6 строчку в Apple Music и чарте ВКонтакте,
затем добрался до 2-го места в “Яндекс Музыке”. В 2023-м поп-певица записала совместный
трек “Королева” с Филиппом Киркоровым. Также долгое время в российских чартах первые
строчки занимали такие композиции Люси, как “Аэроэкспресс”, “Плакал Голливуд”. </p>
</div>
<div class="item"> <h2>Песня “Солнце Монако”</h2> <p>Зачем мне солнце Монако.
Для чего, скажи мне, луна Сен-Тропе. Когда твой взгляд светит ярко. В этом смысла ноль,
если тебя рядом нет. </p></div> </body>

```

Реализация данного кода представлена на рисунке 9.12.

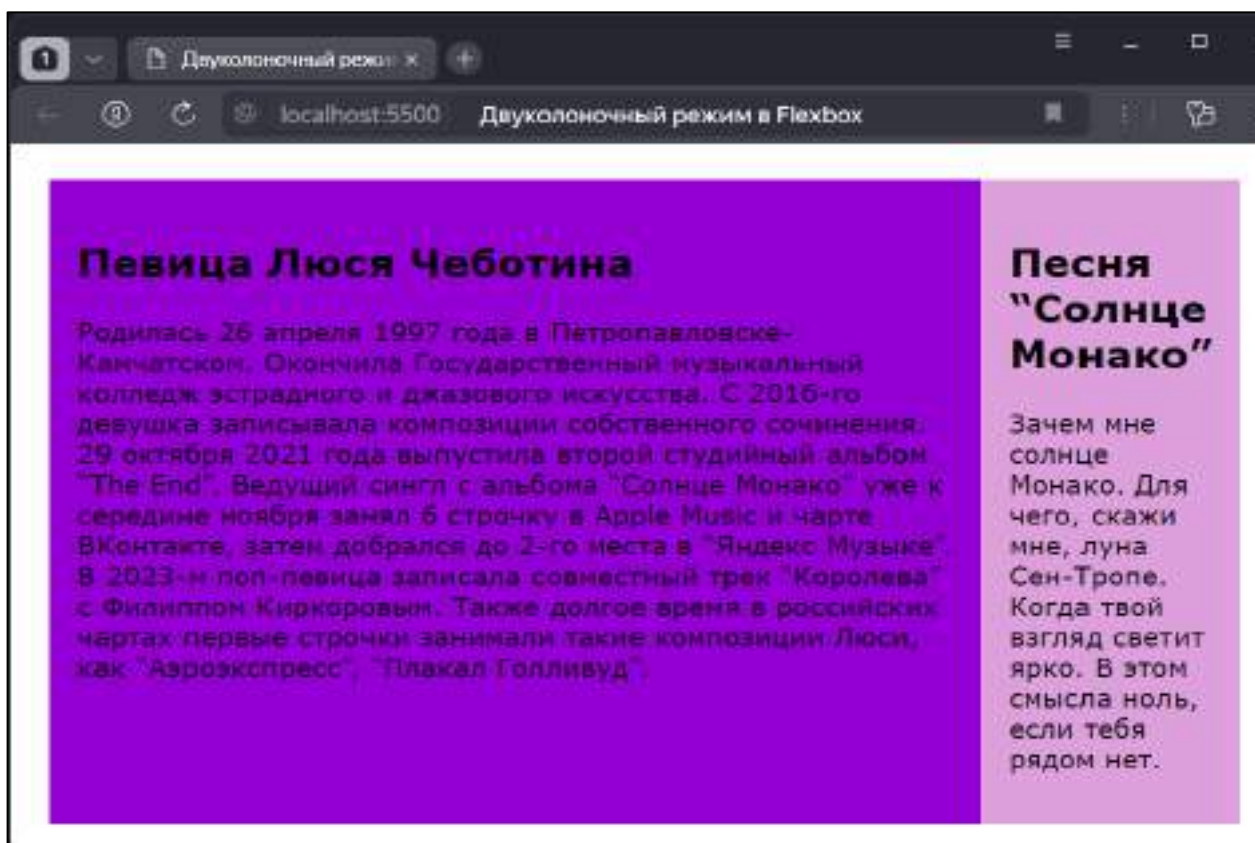


Рисунок 9.12 - Двуколоночный режим в Flexbox

Здесь flex-контейнером является элемент body. Так как на мобильных устройствах (особенно смартфонах) размер экрана не такой большой, поэтому по умолчанию устанавливается расположение элементов в столбик. Однако для устройств с экраном от 600px и выше действует правило media-query, которое устанавливает расположение в виде строки.

9.6.2 Трехколоночный режим

Далее представлен пример создания гибкого макета веб-приложения, в котором заданы три колонки.

```
<head> <style>
    *{ box-sizing: border-box; }
    html, body { padding: 1; margin: 1; font-family: verdana, arial, sans-serif;}
    body {display: flex; padding: 1.1em; flex-direction: column; }
    .item {background-color: PaleTurquoise; font-size: 1.1em; padding: 1em; flex: 1;}
    .item:nth-child(1) { background-color: DarkTurquoise;}
    @media screen and (min-width: 600px) { body {flex-direction: row;}
    .item:nth-child(2) {order: -1;}}
```

```
<body> <div class="item">
```

```
    <h1>Певец Дима Билан</h1> <p>Сверхпопулярный поп-исполнитель, первый
    российский певец, победивший в конкурсе “Евровидение” - Дима Билан – родился в 1981-м
    году в Карачаево-Черкессии. С ранних лет мальчик тянулся к творчеству, он с большим
    удовольствием учился в музыкальной школе, участвовал в городских конкурсах и
    фестивалях. </p> </div>
```

```
    <div class="item"> <h3>Награды певца</h3>
```

```
    <p>Дима Билан является многократным лауреатом различных музыкальных наград, в
    числе которых премии “МУЗ-ТВ”, RU.TV, ZD Awards, MTV ЕМА, телевизионный фестиваль
    “Песня года” и другие. 14-кратный обладатель национальной музыкальной премии “Золотой
    граммофон” радиостанции “Русское Радио”. Рекордсмен по количеству наград премий MTV
    RMA и “МУЗ-ТВ”. </p> </div>
```

```
    <div class="item">
```

```
    <h3>Топ-10 лучших песен артиста</h3>
```

```
    <ul> <li>“Ночной хулиган”</li>
```

```

<li>“Believe”</li>
<li>“Я тебя помню”</li>
<li>“Невозможное возможно”</li>
<li>“Это была любовь” </li>
<li>“На берегу неба” </li>
<li>“Неделимые” </li>
<li>“Малыш”” </li>
<li>“Молния” </li>
<li>“Ты должна рядом быть” </li></ul> </p> </div> </body>

```

Реализация данного кода представлена на рисунке 9.13.

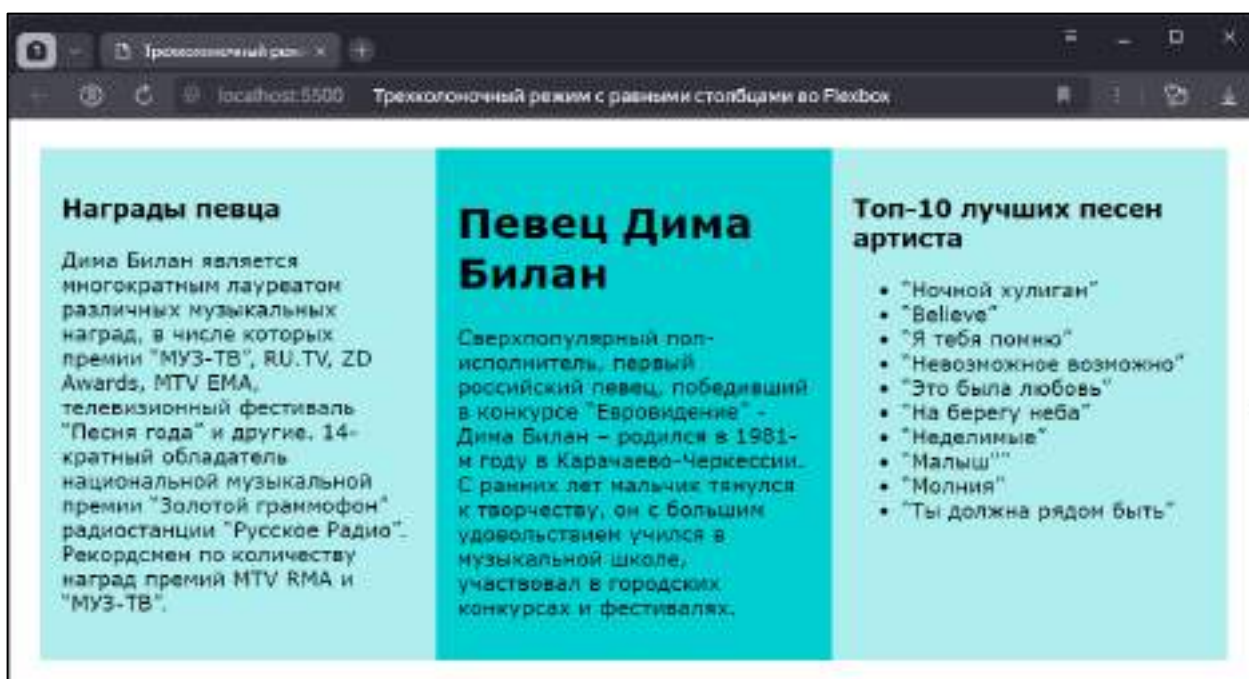


Рисунок 9.13 - Трехколоночный режим с равными столбцами во Flexbox

В отличие от предыдущего примера здесь добавлен еще один элемент. Особенностью этого примера является то, что столбцы имеют одинаковые размеры. Для этого у них установлено свойство `flex: 1`, то есть при растяжении или уменьшении границ контейнера все элементы будут масштабироваться на равную величину.

И кроме того, при ширине экрана больше 600px у второго элемента устанавливается свойство `order: -1`, благодаря чему этот элемент помещается первым.

Подобным образом можно добавить и большее количество столбцов. Но в данном случае по умолчанию столбцы имеют одинаковую ширину. В том случае, если

один из столбцов должен иметь ширину больше, чем у остальных добавляется в стили страницы следующее правило:

```
.item:first-child { flex: 0 0 50%;}
```

В этом случае первый элемент всегда будет занимать 50% пространства контейнера (рисунок 9.14):

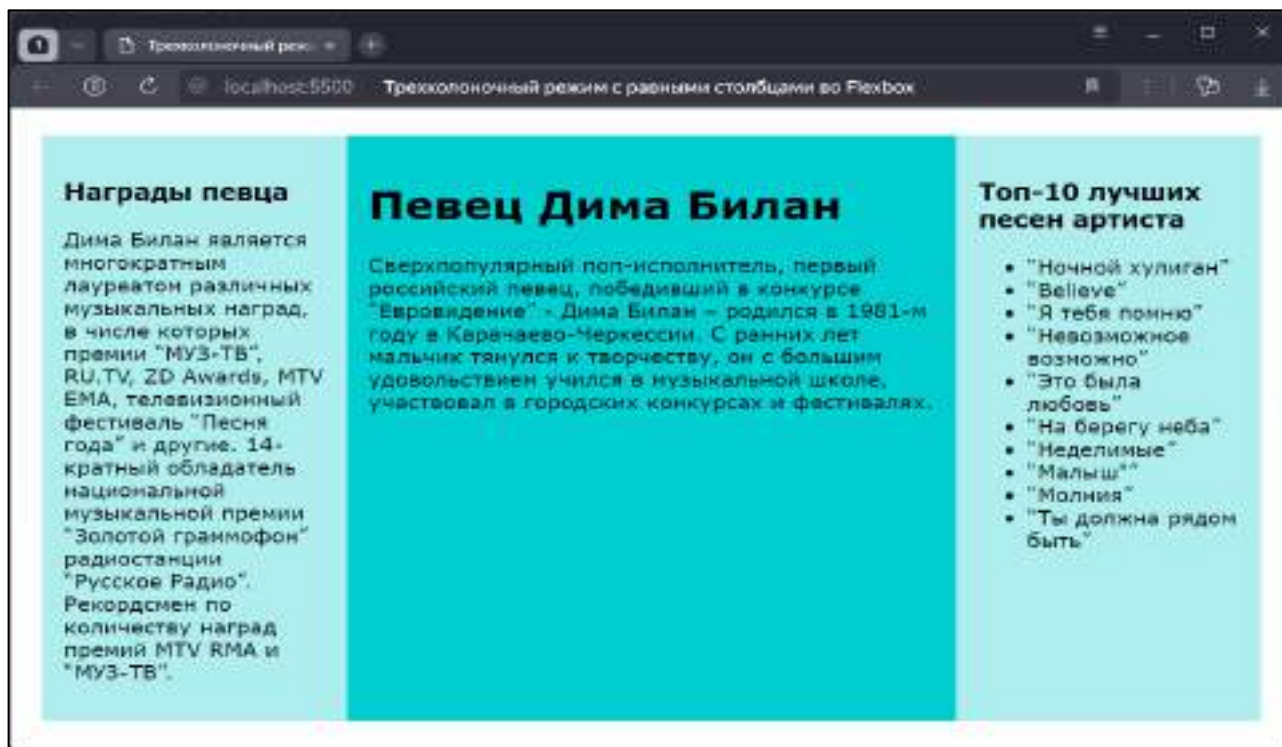


Рисунок 9.14 - Трехколоночный дизайн в Flexbox и CSS3

9.7 Пример макета страницы на Flexbox

Далее приведен приме создания гибкого макета страницы, состоящего из шапки, футера и центральной части, в которой задано три столбца: основное содержимое и два сайдбара. Код веб-страницы представлен ниже.

```
<head> <style>
```

```
* { box-sizing: border-box; }
```

```
html, body {padding: 1; margin: 1; font-family: verdana, arial, sans-serif; }
```

```
body {color:white;
```

```

    font-size: 1.2em;
    padding: 1.1em;
    display: flex;
    flex-direction: column;}
main {display: flex; flex-direction: column;}
article {background-color: LimeGreen; flex: 2 2 12em; padding: 1em; }
nav,aside {flex: 1; background-color: YellowGreen ;}
    nav { order: -1;}
header, footer { flex: 0 0 5em; background-color: SeaGreen;}
@media screen and (min-width: 600px) { body {min-height: 100vh; }
main {flex-direction: row; min-height: 100%; flex: 1 1 auto;}} </style></head>
<body> <header> <h1>Золотой голос России - Муслим Магомаев</h1>
<h2> Веб-страница, посвященная творчеству Муслима Магомаева<h2> </header>
<main> <article> <h1> Муслим Магометович Магомаев </h1>
    <p> Советский, азербайджанский и российский эстрадный и оперный певец,
композитор, киноактёр. Народный артист СССР. За время карьеры достиг популярности как
в странах бывшего СССР, так и других странах Восточной Европы и Центральной Азии.
</p></article>
    <nav> <h2>Лучшие песни</h2> </nav>
    <a href=azerb.html>“Азербайджан”</a>
    <a href=verni.html>“Верни мне музыку”</a>
    <a href=poimi.html>“Пойми меня” </a>
    <a href=cvet.html>“Цветные сны”</a>
    <a href=melod.html>“Мелодия”</a>
    <a href=podm.html>“Подмосковные вечера”</a>
    <a href=padaed.html>“Падает снег”</a>
    <a href=luchshii.html>“Лучший город земли”</a>
    <a href=koroleva.html>“Королева красоты”</a>
    <a href=siniya.html>“Синяя вечность”</a>
    <aside><h2>Семья артиста</h2> <p>Муслим Магомаев родился 17 августа 1942 года в
Баку. Отец, Магомет Магомаев, погиб на фронте, не дожив до Победы 15 дней. Мама Муслима
Айшет Магомаева - драматическая актриса. Муслим считал себя по национальности
азербайджанцем, а Россию - матерью. В детстве Муслим Магомаев рос в семье дяди Джамала
Муслимовича. </p> </aside> </main>

```


`<footer> <p>Последние годы жизни он прожил в Москве, отказываясь от концертных выступлений. Занимался живописью, вёл переписку с поклонниками через свой веб-сайт.</p>`
`</footer> </body>`

Реализация данного кода представлена на рисунке 9.15.

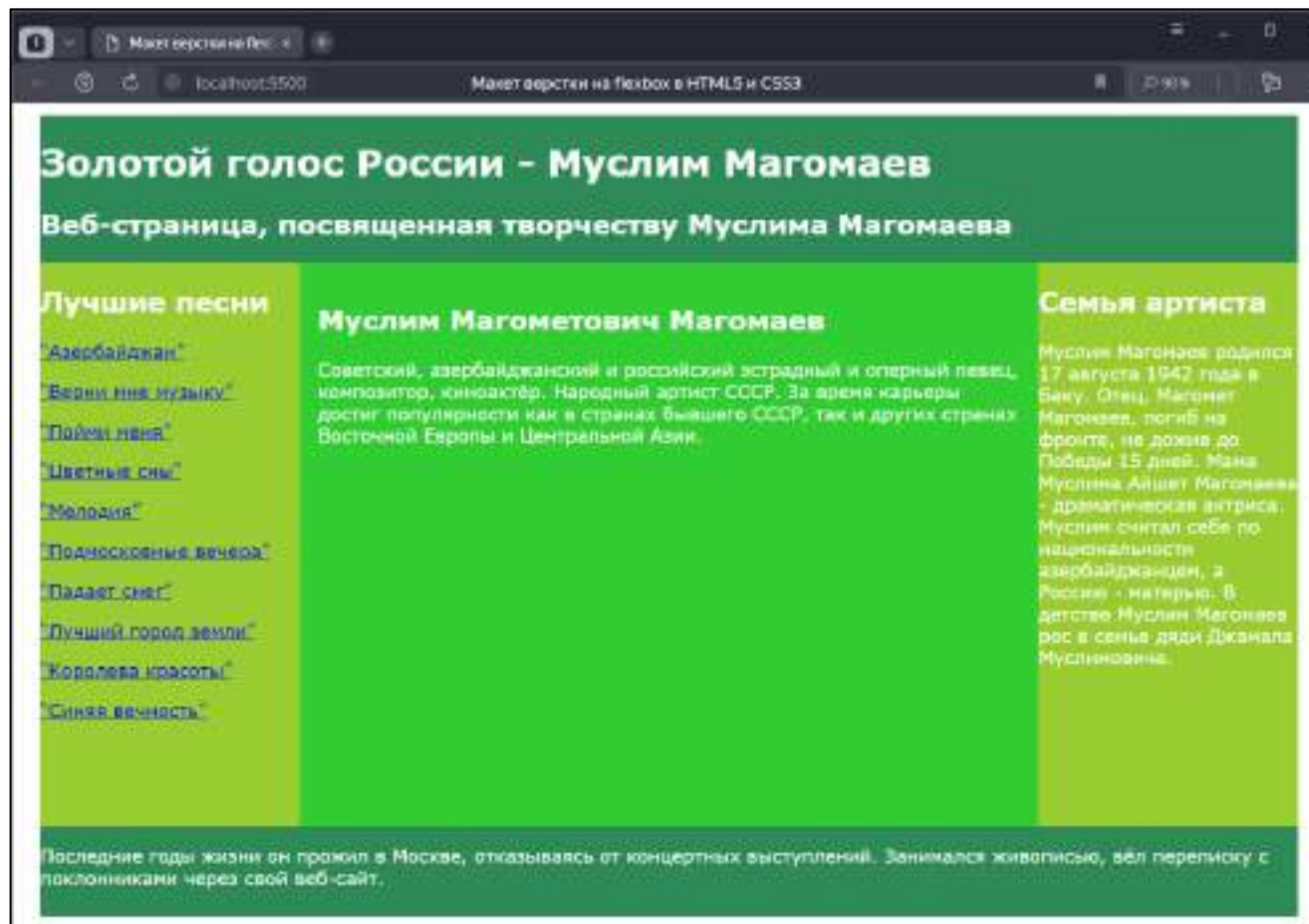


Рисунок 9.15 - Макет верстки на flexbox в HTML5 и CSS3

В данном примере, flex-контейнером верхнего уровня здесь является элемент `body`. Его flex-элементами являются `header`, `main` и `footer`. `Body` располагает все свои элементы сверху вниз в столбик. Здесь также стоит отметить, что при ширине от 600px и выше для заполнения всего пространства браузера у `body` устанавливается стиль `height: 100vh`; Элементы `header` и `footer` аналогичны. Их свойство `flex: 0 0 5em`; указывают, что при любом изменении контейнера эти элементы будут иметь размер в 5em. То есть они имеют статический размер.

Более сложным является элемент `main`, который определяет основное содержимое. При этом, будучи flex-элементом, он также является flex-контейнером

для вложенных элементов и управляет их позиционированием. При ширине браузера до 600px он располагает элементы в столбик, что очень удобно на мобильных устройствах. При ширине от 600px вложенные элементы nav, article и aside располагаются в виде строки. И поскольку при такой ширине браузера родительский элемент body заполняет по высоте все пространство браузера, то для заполнения всей высоты контейнера body при его изменении у элемента main устанавливается свойство flex: 1 1 auto;

У вложенных в main flex-элементов стоит отметить, что элемент навигации nav и элемент сайдбара aside будут иметь одинаковые размеры при масштабировании контейнера. А элемент article, содержащий основное содержимое, будет соответственно больше. При этом хотя nav определен после элемента article, но благодаря установке свойства order: -1 блок навигации будет стоять до блока article.

9.8 Контрольные вопросы

- 1) Понятие Flexbox. Описать содержимое контейнера Flex.
- 2) Рассказать о направлениях flex-direction, flex-flow. Порядок элементов.
- 3) Рассказать о выравнивании строк и столбцов. align-content.
- 4) Рассказать об управлении элементами. flex-basis, flex-shrink и flex-grow.
- 5) Рассказать, как создается макет страницы на Flexbox.

9.9 Тестирование по главе

- 1) Начало центральной оси описывает термин ...
 - a) main start;
 - б) main end;

- в) cross start;
- г) cross end.

2) Свойство ... определяет, будет ли контейнер иметь несколько рядов, чтобы вместить все элементы

- а) flex-direction;
- б) flex-wrap;
- в) flex-flow;
- г) flex-basis.

3) Свойство ... позволяет установить группу, позволяя переопределить его позицию

- а) order;
- б) orphans;
- в) justify-items;
- г) flex-shrink.

4) При указании ... элементы равным образом распределяют пространство между левым и правым краями контейнера

- а) space-between;
- б) space-around;
- в) space-evenly;
- г) stretch.

5) При указании значения ... для align-content строки занимают все свободное место

- а) space-between;
- б) space-around;
- в) space-evenly;
- г) stretch.

б) Свойство ... определяет, как элемент будет уменьшаться относительно других элементов в контейнере

- а) flex-basis;
- б) flex-shrink;
- в) flex-grow;
- г) flex-flow.

7) Свойство ... определяет начальный размер контейнера

- а) flex-basis;
- б) flex-shrink;
- в) flex-grow;
- г) flex-flow.

8) Укажите правильную последовательность в использовании свойства flex

- а) flex: [flex-basis] [flex-grow] [flex-shrink];
- б) flex: [flex-grow] [flex-basis] [flex-shrink];
- в) flex: [flex-grow] [flex-shrink] [flex-basis];
- г) flex: [flex-basis] [flex-shrink] [flex-grow].

9) ... - значение по умолчанию свойства flex-direction

- а) row;
- б) row-reverse;
- в) column;
- г) column-reverse.

10) ... - значение по умолчанию свойства flex-wrap

- а) wrap-reverse;
- б) wrap;
- в) nowrap;
- г) unset.

10 Двумерная система сеток Grid Layout

10.1 Определение Grid Layout. Grid Container

Grid Layout представляет специальный модуль CSS3, который позволяет позиционировать элементы в виде сетки или таблицы. Как и Flexbox, Grid Layout представляет гибкий подход к компоновке элементов, только если Flexbox размещает вложенные элементы в одном направлении - по горизонтали в виде столбиков или по вертикали в виде строк, то Grid позиционирует элементы сразу в двух направлениях - в виде строк и столбцов, образуя тем самым таблицу.

Основой для определения компоновки Grid Layout является grid container, внутри которого размещаются элементы. Для создания grid-контейнера необходимо присвоить его стилевому свойству display одно из двух значений: grid или inline-grid.

Далее представлен пример веб-страницы, в которой применяется Grid Layout. В данном примере для контейнера grid-container установлено свойство display:grid. В нем располагается пять grid-элементов.

```
<head> <style>
.container {border: solid 3px black; display: grid; color: black;}
.item { text-align:center;font-size: 1.2em;padding: 1.4em; color: white;}
.col1 {background-color: MediumVioletRed;}.col2 {background-color: DeepPink;}
.col3 {background-color: HotPink;}.col4 {background-color:PaleVioletRed;}
</style></head>
<body> <h3>Российские императоры</h3>
<div class="container">
  <div class="item col1">Пётр I Великий. Годы правления: с 22 октября 1721 - 28 января
1725гг.) </div>
  <div class="item col2">Екатерина I. Годы правления: с 28 января 1725 - 6 мая 1727гг.</div>
  <div class="item col3"> Анна Иоанновна. Годы правления: с 4 февраля 1730 - 17 октября
1740гг. </div>
```

```
<div class="item col4"> Елизавета Петровна. Годы правления: с 25 ноября 1741 - 25 декабря 1761гг. </div>
```

```
<div class="item col1">Екатерина II. Великая. Годы правления: 28 июня 1762 – 6 ноября 1796</div>
```

```
</div> </body>
```

Реализация данного кода представлена на рисунке 10.1.

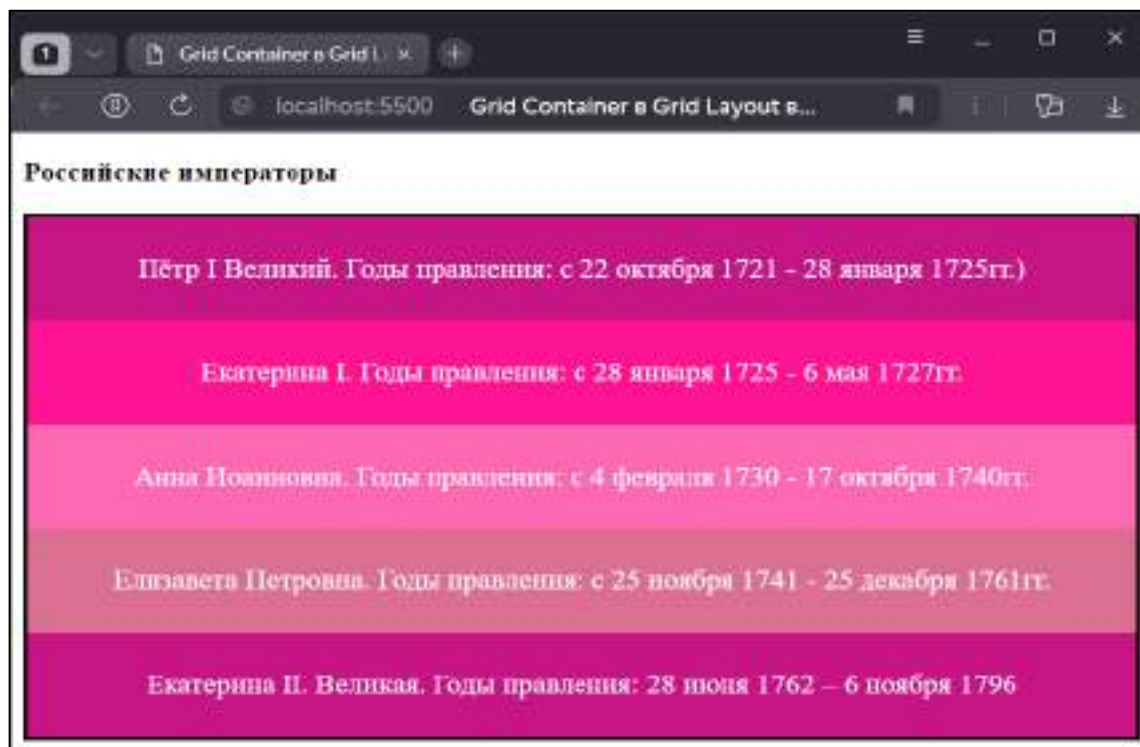


Рисунок 10.1 - Grid Container в Grid Layout в CSS3

Если значение `grid` определяет контейнер как блочный элемент, то значение `inline-grid` определяет элемент как строчный (`inline`). Далее представлен пример применения этого свойства. В этом случае весь грид занимает только то пространство, которое необходимо для размещения его элементов.

```
<head> <style>
.container {border: solid 3px black;display: inline-grid;}
.item {box-sizing: border-box; text-align:center;font-size: 1.2em;
padding: 1.4em; color: black;}
.col1 {background-color: OrangeRed;} .col2 {background-color: Orange;}
.col3 {background-color: Gold ;} .col4 {background-color: Yellow;}
</style></head><body>
```

```

<h3> Театры Оренбурга</h3><div class="container">
  <div class="item col1"> Оренбургский государственный областной драматический театр
им. М. Горького</div>
  <div class="item col2">Оренбургский театр музыкальной комедии</div>
  <div class="item col3">Государственный татарский драматический театр им. Мирхайдара
Файзи</div>
  <div class="item col4"> Оренбургский государственный областной театр кукол</div>
  <div class="item col1">Театр кукол “Пьеро” </div>
</div></body>

```

Реализация данного кода представлена на рисунке 10.2.

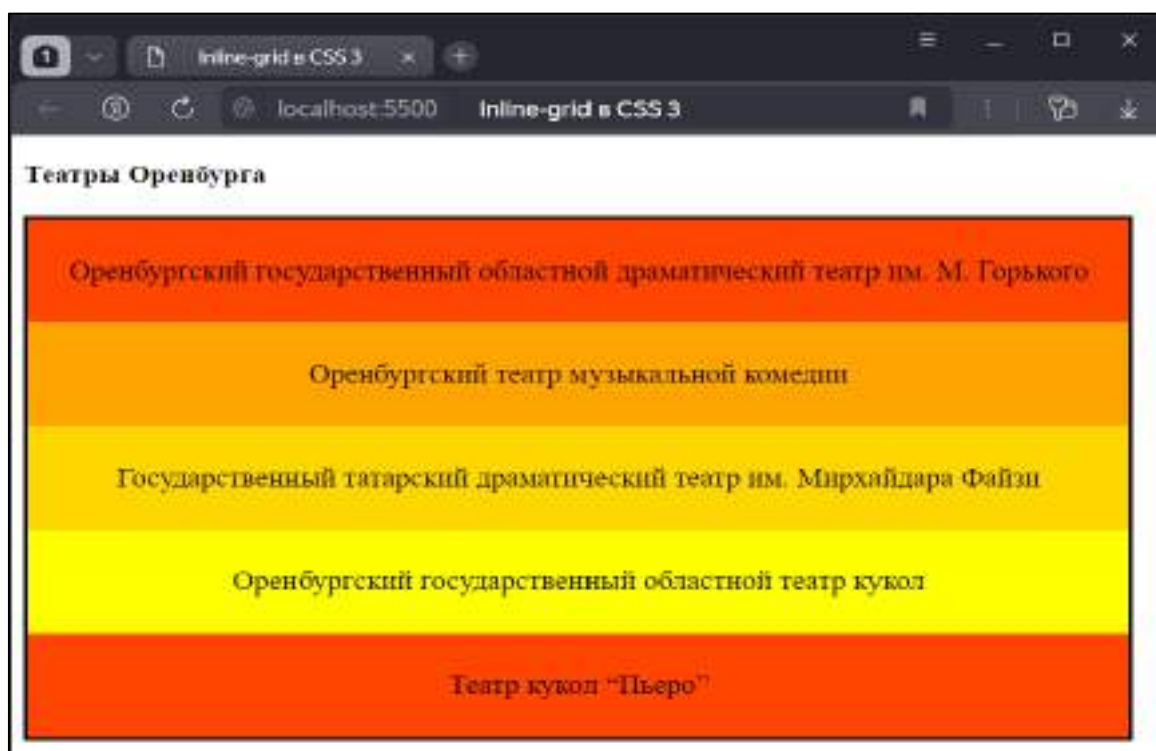


Рисунок 10.2 – Применение Inline-grid в CSS3

10.2 Строки и столбцы

Грид образует сетку из строк и столбцов, на пересечении которых образуются ячейки. И для установки строк и столбцов в Grid Layout использовать следующие свойства CSS3:

- grid-template-columns: настраивает столбцы;
- grid-template-rows: настраивает строки.

Далее представлен пример, где определяется грид с двумя столбцами.

```
<head> <style>
.container {border: solid 3px black;
display: grid;
grid-template-columns: 30em 30em;}
.item { box-sizing: border-box;
text-align:center;
font-size: 1.2em;
padding: 1.4em;
color: white;}
.col1 {background-color: Violet }.col2 {background-color: Magenta;}
.col3 {background-color: MediumOrchid;;}.col4 {background-color: BlueViolet;}
.col5 {background-color: DarkViolet;}
</style> </head> <body>
```

```
<h2>Марки автомобилей и их фирмы-производители</h2>
```

```
<div class="container">
```

```
  <div class="item col1">Форд. Ford Motor Company - американская
автомобилестроительная компания. Четвёртый в мире производитель автомобилей по
объёму выпуска за весь период существования. </div>
```

```
  <div class="item col2"> Мазерати. Maserati S.p.A.-итальянская компания, производитель
эксклюзивных автомобилей спортивного и бизнес-класса. Основана 1 декабря 1914 года в
Болонье, Италия. Штаб-квартира находится в городе Модена, Италия.</div>
```

```
  <div class="item col3"> Ауди. Audi AG - немецкая автомобилестроительная компания в
составе концерна Volkswagen Group, специализирующаяся на выпуске автомобилей под
маркой Audi. Штаб-квартира расположена в городе Ингольштадт, Германия. </div>
```

```
  <div class="item col4"> Мазда. Mazda Motor Corporation-японская
автомобилестроительная компания, выпускающая автомобили “Мазда”. Штаб-квартира
расположена в посёлке Футю, уезд Аки, префектура Хиросима, Япония. </div>
```

```
  <div class="item col5">Пежо. Peugeot - французский производитель автомобилей, с 1976
года входил в группу PSA Peugeot Citroen, а с 2021 года принадлежит корпорации Stellantis.
</div> </div> </body>
```

Реализация данного кода представлена на рисунке 10.3.

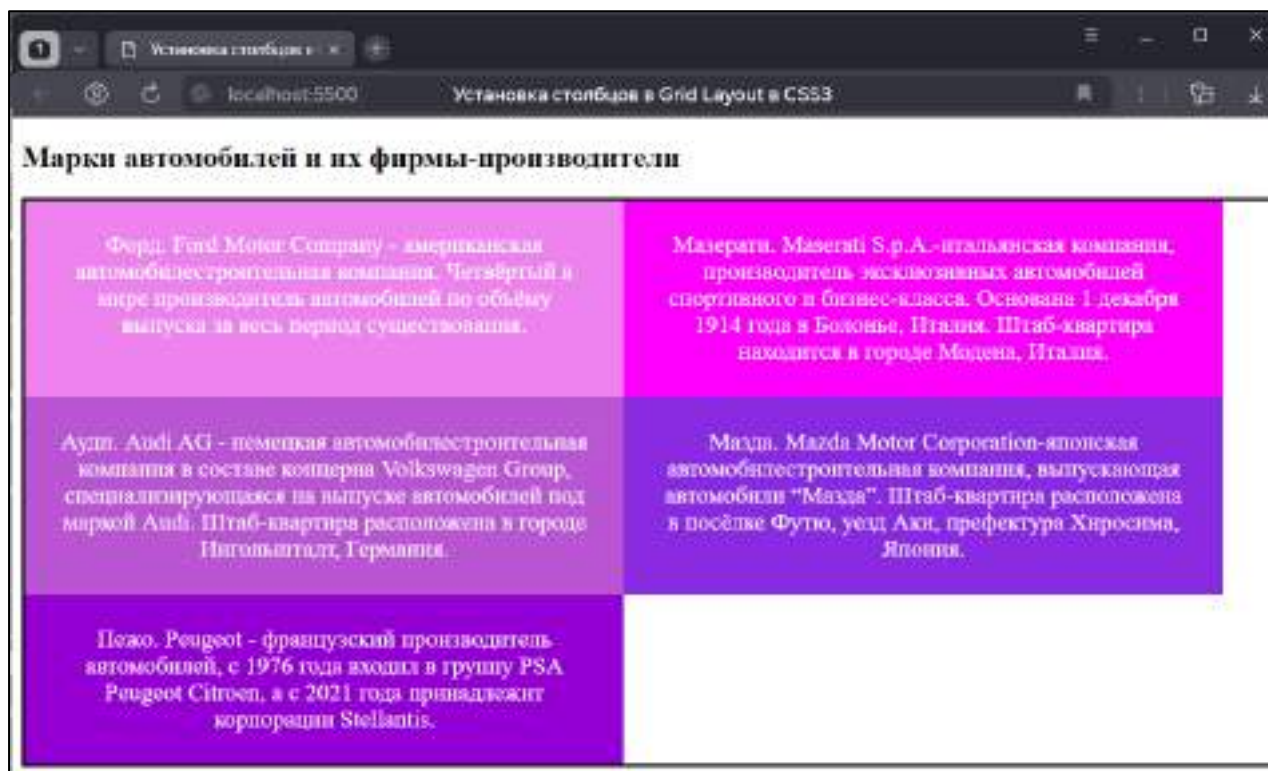


Рисунок 10.3 - Установка столбцов в Grid Layout в CSS3

В качестве значения свойству `grid-template-columns` передается ширина столбцов. Сколько нужно иметь в гриде столбцов, столько и нужно передать значений этому свойству. Так, в случае выше грид содержит два столбца, поэтому свойству передаются два значения, которые указывают ширину столбцов:

Соответственно если необходимо, чтобы в гриде было три столбца, то надо передать три значения, например:

`grid-template-columns: 8em 7em 8em;`

Если столбцов больше, чем элементов, то по умолчанию для их вмещения создаются новые строки.

Настойка строк во многом аналогичная настройке столбцов. Для этого у грид-контейнера необходимо установить свойство `grid-template-rows`, которое задает количество и размеры строк.

Далее представлен пример применения свойства `grid-template-rows`. Свойству `grid-template-rows` передается высота каждой из строк. Так, в данном случае высота первой строки составляет `5em`, а второй - `8em`.

```

<head> <style> .container {border: solid 3px black; display: grid; grid-template-columns: 25em
25em 25em; grid-template-rows: 12em 16em;}
    .item {text-align:center;font-size: 1.2em; padding: 1.4em;}
    .col1 {background-color: DarkCyan;} .col2 {background-color: LightSeaGreen; }
    .col3 {background-color: DarkTurquoise;} .col4 {background-color: MediumAquaMarine;}
    .col5 {background-color: Cyan;} .col6 {background-color: PaleTurquoise;}
    .col7 {background-color: Aquamarine;} </style></head>
<body> <h3> Жанры изобразительного искусства. </h3> <div class="container">
    <div class="item col1">Пейзаж - жанр изобразительного искусства (а также отдельные
произведения этого жанра), в котором основным предметом изображения является
первозданная либо в той или иной степени преображённая человеком природа. </div>
    <div class="item col2">Портрет - это жанр изобразительного искусства, посвященный
воспроизведению образа конкретного человека или группы людей, показывающий
индивидуальные черты человека. </div>
    <div class="item col3">Бытовой жанр - это жанр изобразительного искусства, предметом
которого является изображение сцен повседневной, частной и общественной жизни, обычно
современной художнику. </div>
    <div class="item col4">Анималистика - жанр изобразительного искусства, главным
мотивом и основным объектом которого являются животные, главным образом в живописи,
фотографии, скульптуре, графике и реже в декоративном искусстве.</div>
    <div class="item col5">Натюрморт – это жанр изобразительного искусства, в котором
изображаются предметы неодушевлённой природы, соединённые художником в отдельную
группу и представляющие собой целостную композицию либо включённые в композицию
иного жанра.</div>
    <div class="item col6">Мифический жанр - это жанр изобразительного искусства, в
котором основой для сюжета произведений служат древние сказания, легенды, предания или
сказки разных народов. </div>
    <div class="item col7">Батальный жанр – это жанр изобразительного искусства,
посвящённый темам войны и песочницей. Главное место в батальном жанре занимают
сцены сухопутных, морских сражений и военных походов. </div> </div> </body>

```

Реализация данного кода представлена на рисунке 10.4.

В то же время, если элементов больше, чем ячеек грида, то образуются дополнительные строки (как в случае со столбцами). Поэтому, несмотря на то что выше были определены настройки только для двух строк, в реальности строк в гриде

будет три строки, причем, как видно на скриншоте, высота третьей строки необязательно будет 5em, как у других строк, она будет вычисляться автоматически.

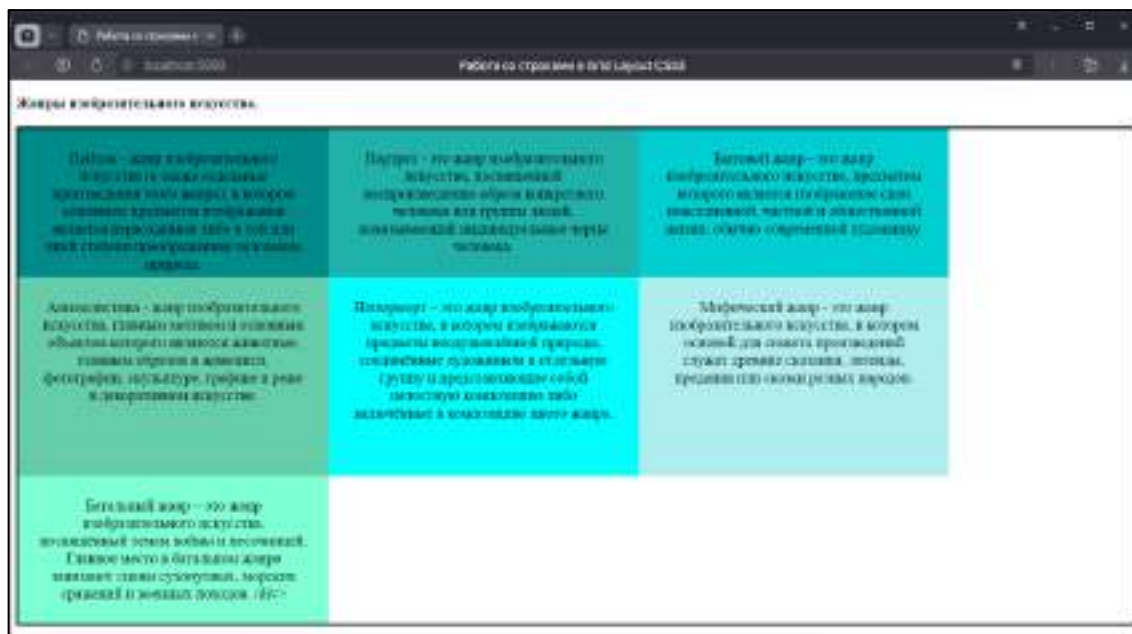


Рисунок 10.4 – Работа со строками в Grid Layout CSS3

Можно изменить стили grid-контейнера, добавив еще две строки:

```
.container {border: solid 3px black;display: grid; grid-template-columns: 25em 25em 25em; grid-template-rows: 12em 12em 12em 12em;}
```

В результате будут сформированы три столбца и четыре строки (рисунок 10.5).

И поскольку ячеек грида больше, чем элементов, последняя строка пустая.

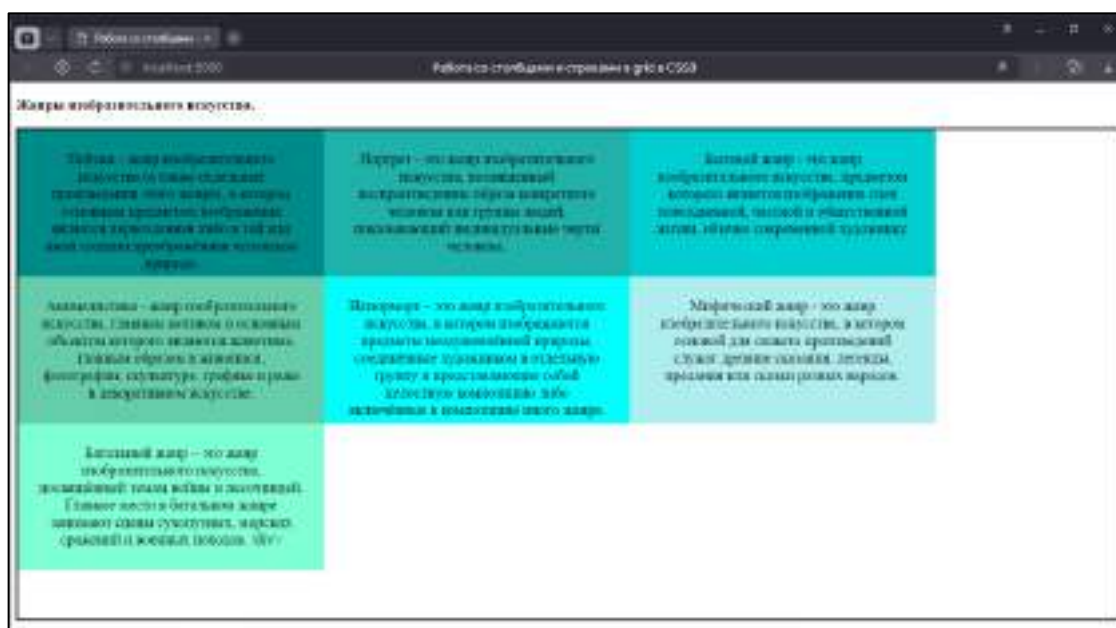


Рисунок 10.5 – Работа со столбцами и строками в grid в CSS3

10.3 Функция repeat и свойство grid

10.3.1 Повторение строк и столбцов. Свойство repeat

Если столбцов и(или) строк много и они имеют одинаковые размеры, то есть смысл использовать специальную функцию repeat(), которая позволит настроить строки и столбцы. Так, в примере выше повторяется определение одинаковых строк и столбцов в grid-контейнере [1]:

```
grid-template-columns: 8em 8em 8em;  
grid-template-rows: 5em 5em 5em 5em;
```

Здесь происходит повторение одних и тех же размеров - 8em и 5em для установки ширины столбцов и высоты строк. Поэтому переписываются стили, применив функцию repeat:

```
.container { border: solid 3px black; display: grid; grid-template-columns: repeat(3, 8em);  
grid-template-rows: repeat(4, 5em); }
```

Первый параметр функции repeat представляет число повторений, а второй - определение строк или столбцов. Например, свойство grid-template-columns: repeat(3, 8em); говорит, что необходимо определить 3 столбца шириной в 8em.

Соответственно выражение grid-template-rows: repeat(4, 5em) определяет 4 строки высотой по 5em.

Можно задавать повторение нескольких столбцов и строк:

```
.container { border: solid 3px black; display: grid;  
grid-template-columns: repeat(2, 7em 8em); grid-template-rows: 6em repeat(3, 5em); }
```

В данном случае будет создано 4 столбца: два раза будут повторяться два столбца с шириной 7em и 8em. В случае со строками будет создано 4 строки. Причем первая будет иметь высоту в 6em, а остальные три - 5em.

10.3.2 Свойство grid

Свойство `grid` объединяет свойства `grid-template-rows` и `grid-template-columns` и разом позволяет задать настройки для строк и столбцов в следующем формате:

`grid: grid-template-rows / grid-template-columns;`

К примеру, есть следующее определение класса `grid`-контейнера:

```
.container { border: solid 3px black;  
display: grid; grid-template-columns: 8em 8em 8em;  
grid-template-rows: 5em 5em 5em 5em;}
```

Можно сократить этот класс следующим образом:

```
.container {border: solid 3px black; display: grid; grid: 5em 5em 5em 5em / 8em 8em 8em;}
```

Либо опять же используя функцию `repeat()`, можно еще больше сократить определение грида:

```
.container { border: solid 3px black; display: grid; grid: repeat(4, 5em) / repeat(3, 8em);}
```

10.4 Размеры строк и столбцов

10.4.1 Фиксированные размеры

В примерах, которые были рассмотрены в предыдущих статьях, ширина столбцов и длина строк устанавливались на основании фиксированных значений, которые передаются свойствам `grid-template-columns` и `grid-template-rows`. Для определения размеров можно использовать самые различные единицы измерения, которые доступны в CSS (`px`, `em`, `rem`, `pt`, `%`), например:

```
.container {border: solid 3px black; display: grid;  
grid-template-columns: repeat(3, 200px); grid-template-rows: repeat(3, 4.5em);}
```

10.4.2 Автоматические размеры

Кроме точных размеров можно задавать автоматические размеры с помощью слова `auto`. В этом случае ширина столбцов и высота строк вычисляются исходя из

размеров содержимого. Далее представлен пример, демонстрирующий задание автоматических размеров столбцов и строк.

```
<head> <style>
    .container {border: solid 3px black; display: grid; grid-template-columns: 11em auto auto;
        grid-template-rows: 8em auto;}
    .item {text-align:center; font-size: 1.2em; padding: 1.4em; border: 1px solid black;}
    .col1 {background-color: LightSteelBlue; } .col2 {background-color: PowderBlue;}
    .col3 {background-color: LightBlue;}      .col4 {background-color: SkyBlue;}
    .col5 {background-color: LightSkyBlue;} .col6 {background-color: DeepSkyBlue;}
</style> </head> <body> <h2>Вузы Оренбургской области</h2>
<div class="container">
    <div class="item col1">Оренбургский государственный университет</div>
    <div class="item col2">Оренбургский государственный аграрный университет </div>
    <div class="item col3">Оренбургский государственный институт искусств </div>
    <div class="item col4">Оренбургский государственный медицинский университет</div>
    <div class="item col5">Оренбургский государственный педагогический университет</div>
    <div class="item col6">Бузулукский гуманитарно-технологический институт
Оренбургского государственного университета</div> </div> </body>
```

Реализация данного кода представлена на рисунке 10.6.

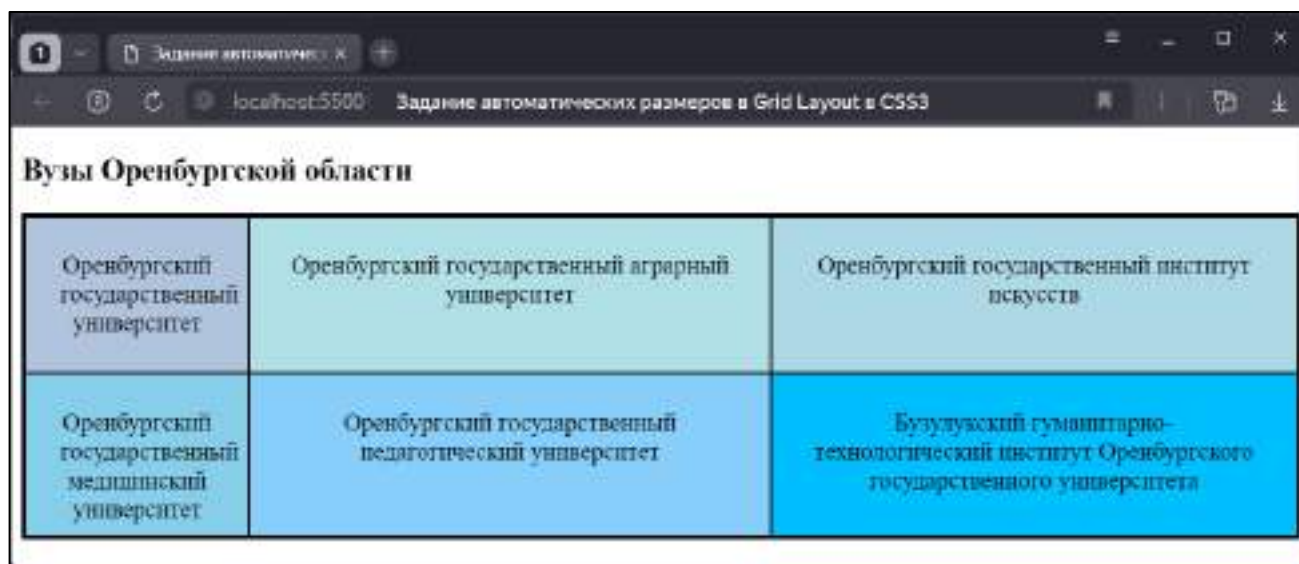


Рисунок 10.6 – Задание автоматических размеров в Grid Layout в CSS3

Как видно на рисунке 10.6, здесь задано две строки и три столбца. Первый столбец имеет фиксированную ширину в 11em, а второй и третий столбцы -

автоматическую ширину. И также первая строка имеют фиксированную высоту, а вторая строка - автоматическую.

10.4.3 Пропорциональные размеры

Для установки пропорциональных размеров применяется специальная единица измерения fr. Она представляет собой часть пространства (fraction), которое отводится для данного столбца или строки. Значение fr еще называют flex-фактором (flex factor). Вычисление пропорциональных размеров производится по формуле:

flex-фактор * доступное_пространство / сумма всех flex-факторов

При этом под доступным пространством понимается все пространство grid-контейнера за исключением фиксированных значений строк и столбцов. Далее приведен пример задания пропорциональных размеров блоков в CSS.

```
<head> <style>
    *{box-sizing: border-box; }
    html, body { margin:1px; padding:1px;}
    .container {height: 80vh; border: solid 3px black;
        display: grid; grid-template-columns: 1.5fr 15em 1fr;
        grid-template-rows: 1fr 12em 1fr;}
    .item {text-align:center; font-size: 1.2em;
        padding: 1.4em;color: white;
        border: 1px solid black;}
    .col1 {background-color: DodgerBlue;}
    .col2{background-color: CornflowerBlue;}
    .col3 {background-color: MediumSlateBlue;}
    .col4 {background-color: RoyalBlue;}
    .col5 {background-color: SteelBlue;}
    .col6 {background-color: CadetBlue;} </style></head>
<body> <h3>Филиалы вузов в городе Оренбурге</h3>
<div class="container">
    <div class="item col1">Оренбургский филиал Российского государственного университета
    нефти и газа им. И.М. Губкина</div>
```

`<div class="item col2">Оренбургский филиал Российского экономического университета имени Г.В. Плеханова</div>`

`<div class="item col3">Оренбургский филиал Российской академии народного хозяйства и государственной службы при Президенте РФ</div>`

`<div class="item col4">Оренбургский институт путей сообщения Самарского государственного университета путей сообщений </div>`

`<div class="item col5">Оренбургский институт (филиал) Московского государственного юридического университета имени О.Е. Кутафина</div>`

`<div class="item col6"> Оренбургский филиал Поволжского государственного университета телекоммуникаций и информатики</div> </div></body>`

Реализация данного кода представлена на рисунке 10.7.



Рисунок 10.7 - Использование Flex factor и fr в Grid Layout и CSS3

В данном случае имеются три столбца с шириной 2fr, 8em, 1fr. Поэтому ширина второго столбца будет вычисляться по формуле:

$$2 * (\text{ширина_грида} - 15\text{em}) / (1.5 + 1)$$

Ширина третьего столбца будет вычисляться по формуле:

$$1 * (\text{ширина_грида} - 15\text{em}) / (1.5 + 1)$$

И если первый столбец, фиксированный с шириной 15em, то ширина второго и третьего столбца будут зависеть от ширины контейнера и будут автоматически масштабироваться при ее изменении. В отношении строк все аналогично.

10.5 Отступы между столбцами и строками

Для создания отступов между столбцами и строками применяются свойства `grid-column-gap` и `grid-row-gap` соответственно. Далее приведен пример задания отступов между строками и столбцами.

```
<head> <style>
  *{box-sizing: border-box;}
  html, body{ margin:1px; padding:1px;}
  .container {height: 80vh; display: grid; grid-template-columns: repeat(3, 1fr);
    grid-template-rows: repeat(3, 1fr); grid-column-gap: 9px; grid-row-gap: 9px;}
  .item {text-align:center; font-size: 1.2em; padding: 1.4em;border: 1px solid black;}
  .col1 {background-color: Beige;} .col2 {background-color: OldLace;}
  .col3 {background-color: AntiqueWhite;} .col4 {background-color: LavenderBlush;}
  .col5 {background-color: MistyRose;} .col6 {background-color: Linen;}
</style> </head>
<body> <h2>Клавишные музыкальные инструменты</h2>
<div class="container">
  <div class="item col1">Фортепиано</div> <div class="item col2">Орган</div>
  <div class="item col3">Рояль</div><div class="item col4">Клавесин</div>
  <div class="item col5">Аккордеон</div><div class="item col6">Фисгармония</div>
</div> </body>
```

Реализация данного кода представлена на рисунке 10.8

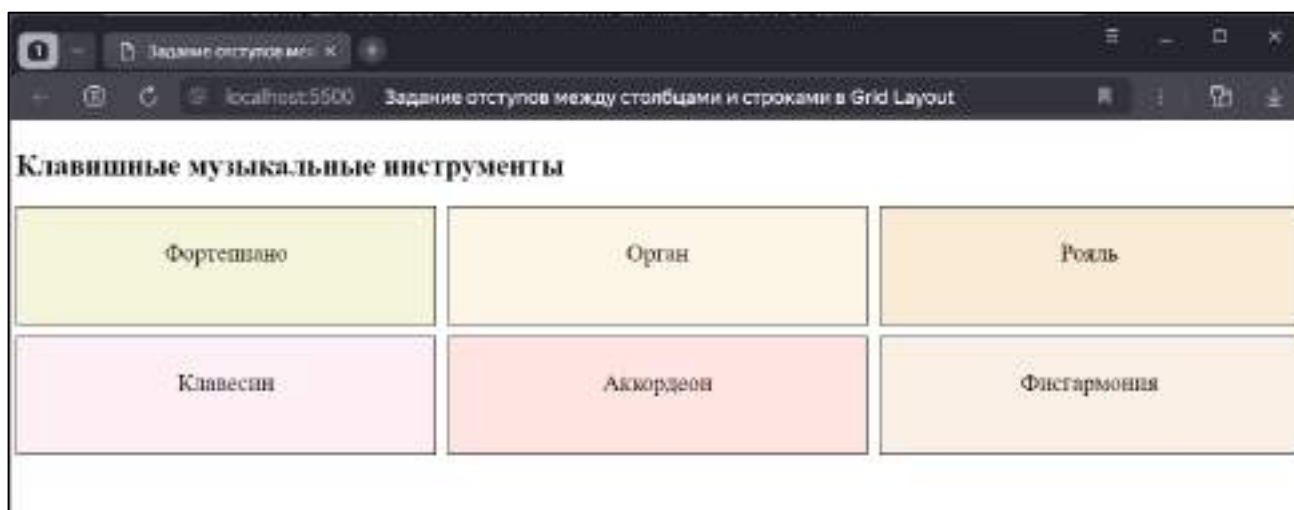


Рисунок 10.8 – Задание отступов между столбцами и строками в Grid Layout

Если значения свойств `grid-column-gap` и `grid-row-gap` совпадают, то вместо них можно определить одно свойство `gap` (ранее назвалось `grid-gap`), которое установит оба отступа:

```
.container {height: 80vh; display: grid; grid-template-columns: repeat(3, 1fr);  
grid-template-rows: repeat(3, 1fr); grid-column-gap: 9px; grid-row-gap: 9px;}
```

10.6 Позиционирование элементов в системе сеток Grid Layout

Грид - это совокупность ячеек, образованных на пересечении столбцов и строк. Но сами строки и столбцы создаются с использованием `grid`-линий, пересекающих сетку по вертикали и горизонтали (рисунок 10.9):

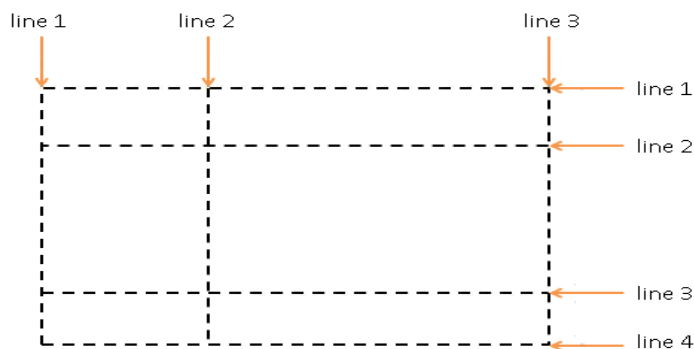


Рисунок 10.9 - Grid-lines in Grid Layout CSS3

По умолчанию каждый элемент сетки располагается в ячейке по очереди. Однако более точно настроить положение элемента в сетке можно с помощью ряда свойств:

- `grid-row-start`: указывает начальную горизонтальную линию сетки, с которой начинается элемент;
- `grid-row-end`: указывает, до какой горизонтальной `grid`-линии следует растягивать элемент;
- `grid-column-start`: определяет вертикальную начальную линию сетки, с которой начинается элемент;

– grid-column-end: указывает, до какой вертикальной линии сетки следует растянуть элемент.

Далее приведен пример растягивания элементов на несколько столбцов.

```
<head> <style>.container { border: solid 3px black;display: grid;
grid-template-columns: repeat(4, 1fr);grid-template-rows: repeat(2, 5em); color:white;}
.s-item{ grid-column-start:2;  grid-column-end: 5; }
.item {text-align:center; font-size: 1.2em; padding: 1.4em ; border:1px solid black;}
.col1 {background-color: Gainsboro;} .col2 {background-color: Silver;}
.col3 {background-color: DarkGrey;} .col4 {background-color: DimGrey;}
.col5 {background-color: SlateGrey ;} .col6 {background-color: DarkSlateGrey;} </style>
</head> <body> <h2> Струнные музыкальные инструменты</h2><div class="container">
<div class="item col1">Гитара</div>
<div class="item col2 s-item">Скрипка</div>
<div class="item col3">Альт</div>
<div class="item col4">Виолончель</div>
<div class="item col5">Контрабас</div>
<div class="item col6">Балалайка</div> </div> </body>
```

Реализация данного кода представлена на рисунке 10.10.

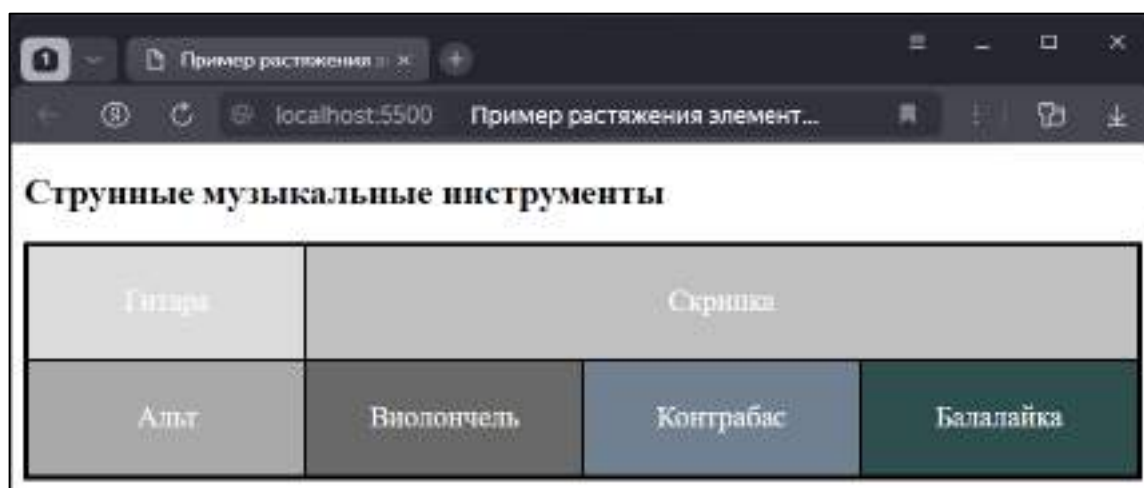


Рисунок 10.10 – Пример растяжения элементов на столбцы в Grid Layout

Здесь определяются четыре столбца, а второму элементу назначается специальный элемент специального класса, который располагается начиная со 2-й линии сетки или 2-го столбца (grid-column-start: 2) до 5-й вертикальной линии сетки (grid-column -end : 5).

Второй элемент не обязательно должен начинаться со второго столбца, он может быть любым другим: первым, третьим и т. д. Например, если размещается второй элемент, начиная с 3-го столбца, на месте второго появится пустота (рисунок 10.11):

```
.s-item{ grid-column-start: 3; grid-column-end: 5;}
```

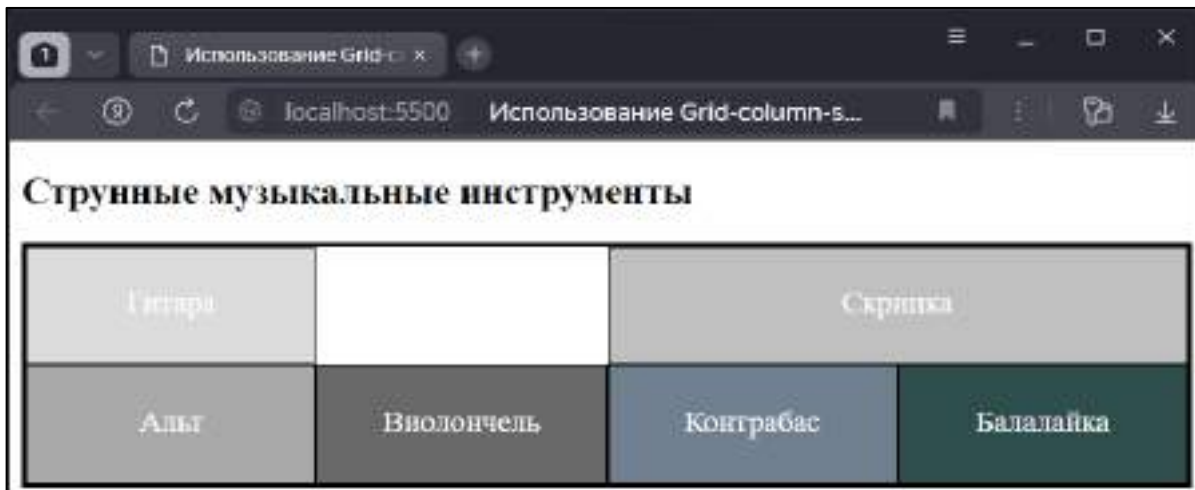


Рисунок 10.11 - Использование Grid-column-start в Grid Layout в CSS3

Если установить второй элемент так, чтобы он начинался в первом столбце, второй элемент будет перемещен в следующую строку и, таким образом, начнется в первом столбце.

Вместо использования двух рассмотренных выше свойств можно использовать одно свойство - `grid-column`, которое принимает значения `grid-column-start` и `grid-column-end`, разделенные косой чертой:

```
grid-column: grid-column-start / grid-column-end;
```

Например, можно сократить стиль класса `special-item` следующим образом:

```
.s-item{grid-column: 3 / 5;}
```

Аналогично с помощью свойств `grid-row-start` и `grid-row-end` можно задать позиционирование элемента на несколько строк. Так, изменим класс `special-item` следующим образом:

```
.s-item{ grid-column-start:2; grid-row-start: 1;grid-row-end: 3;}
```

В данном случае второй элемент позиционируется во втором столбце первой строки и растягивается до 3-й строки (рисунок 10.12).

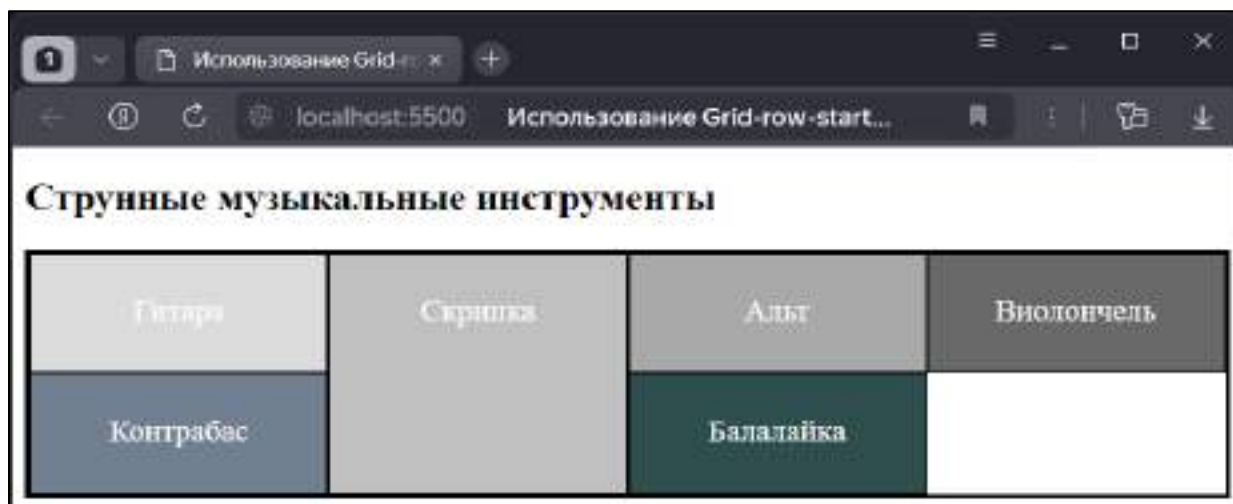


Рисунок 10.12 – Использование Grid-row-start в Grid Layout и CSS3

Вместо использования пары свойств `grid-row-start` и `grid-row-end` можно использовать одно общее свойство `grid-row`:

`grid-row: grid-row-start / grid-row-end;`

Так, можно изменить стиль `special-item` следующим образом:

`.s-item{grid-column-start:2;grid-row: 1 / 3; }`

10.6.1 Растяжение ячеек с помощью свойства Span

С помощью специального слова `span` можно задать растяжение элемента на несколько ячеек. После слова `span` указывается, на какое количество ячеек надо растянуть элемент.

`.s-item{grid-row: 1 / span 2; grid-column: 2 / span 2; }`

Элемент помещается в ячейку, которая находится на пересечении первой строки и второго столбца, и растягивается на две строки вниз и на два столбца вправо.

10.6.2 Grid-area

Свойство `grid-area` объединяет свойства `grid-column` и `grid-row`, позволяя сократить их запись:

grid-area: row-start / column-start / row-end / column-end.

На рисунке 10.13 стили класса `special-item` были изменены:

.s-item{grid-area: 1 / 2 / 3 / 4; }

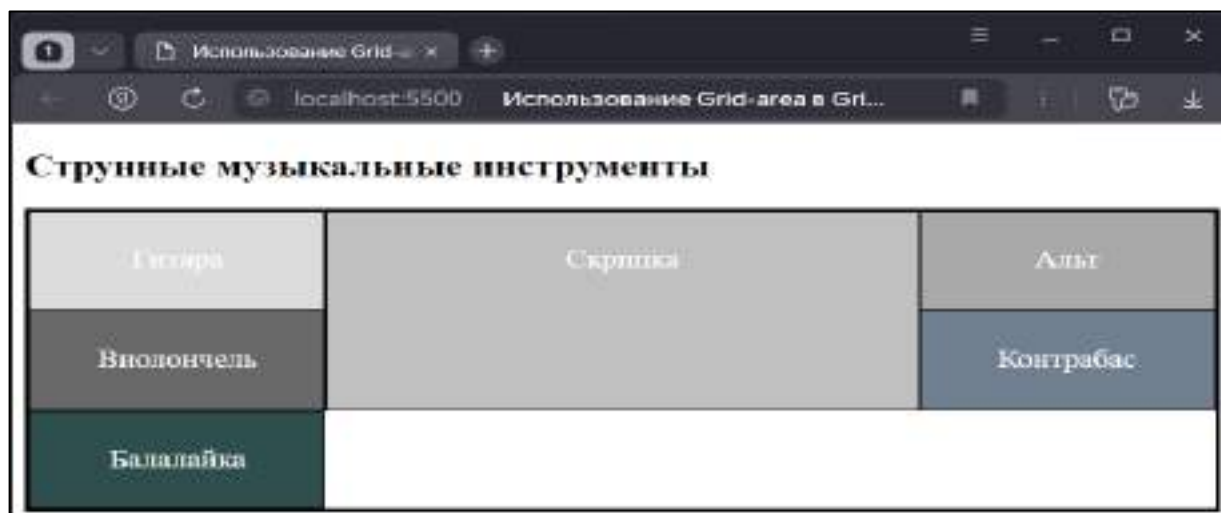


Рисунок 10.13 – Использование Grid-area в Grid Layout в CSS3

10.7 Направление и порядок элементов

10.7.1 Grid-auto-flow

По умолчанию все элементы располагаются по порядку горизонтально, если места в строке больше нет, то элементы переносятся на следующую строку. Но с помощью свойства `grid-auto-flow` можно изменить направление элементов. Это свойство принимает два значения:

- `row`: значение по умолчанию, элементы располагаются в строку друг за другом, если места в строке не хватает, элементы переносятся на следующую строку;
- `column`: элементы располагаются в столбик, если места в столбце не хватает, то элементы переходят в следующий столбец.

Далее приведен пример применения данного свойства.

<head> <style>

```

    *{ box-sizing: border-box;}
html, body{ margin:1px; padding:1px;}
.container { height: 80vh; display: grid;
    grid-template-columns: repeat(3, 1fr);
    grid-template-rows: repeat(2, 1fr);
    grid-auto-flow: row; color: white;}
.item {text-align:center; font-size: 1.2em;
padding: 1.4em;}
.col1 {background-color:Purple;}
.col2 {background-color: Maroon;}
.col3 {background-color: Olive;}
.col4 {background-color: Green;}
.col5 {background-color: Teal;}
.col6 {background-color: Navy;}
</style></head>
<body><h3> Лучшие курорты Краснодарского края</h3>
<div class="container">
    <div class="item col1">Сочи</div>
    <div class="item col2">Анапа</div>
    <div class="item col3">Адлер</div>
    <div class="item col4">Геленджик</div>
    <div class="item col5">Туапсе</div>
    <div class="item col6">Сухуми</div></div></body>

```

Реализация данного кода представлена на рисунке 10.14.

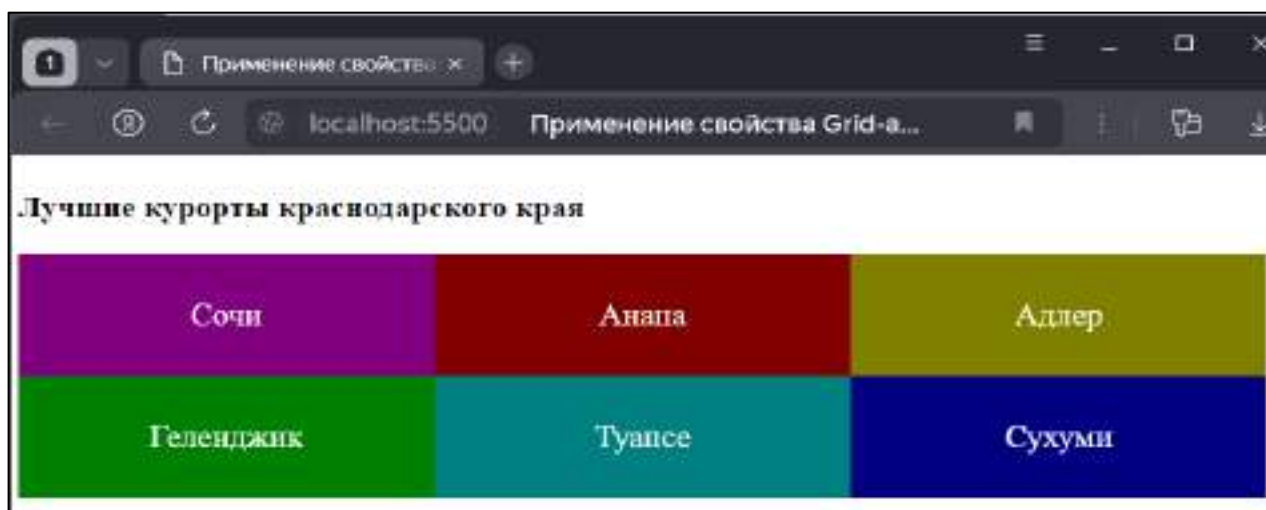


Рисунок 10.14 – Применение свойства Grid-auto-flow в CSS3

Свойство `grid-auto-flow` имеет значение `row`, поэтому элементы будут располагаться в строку. Теперь изменим стиль `grid`-контейнера:

```
.container {height: 80vh; display: grid;  
            grid-template-columns: repeat(3, 1fr);  
            grid-template-rows: repeat(3, 1fr);  
            grid-auto-flow: column; }
```

После этого элементы будут располагаться в столбец (рисунок 10.15).

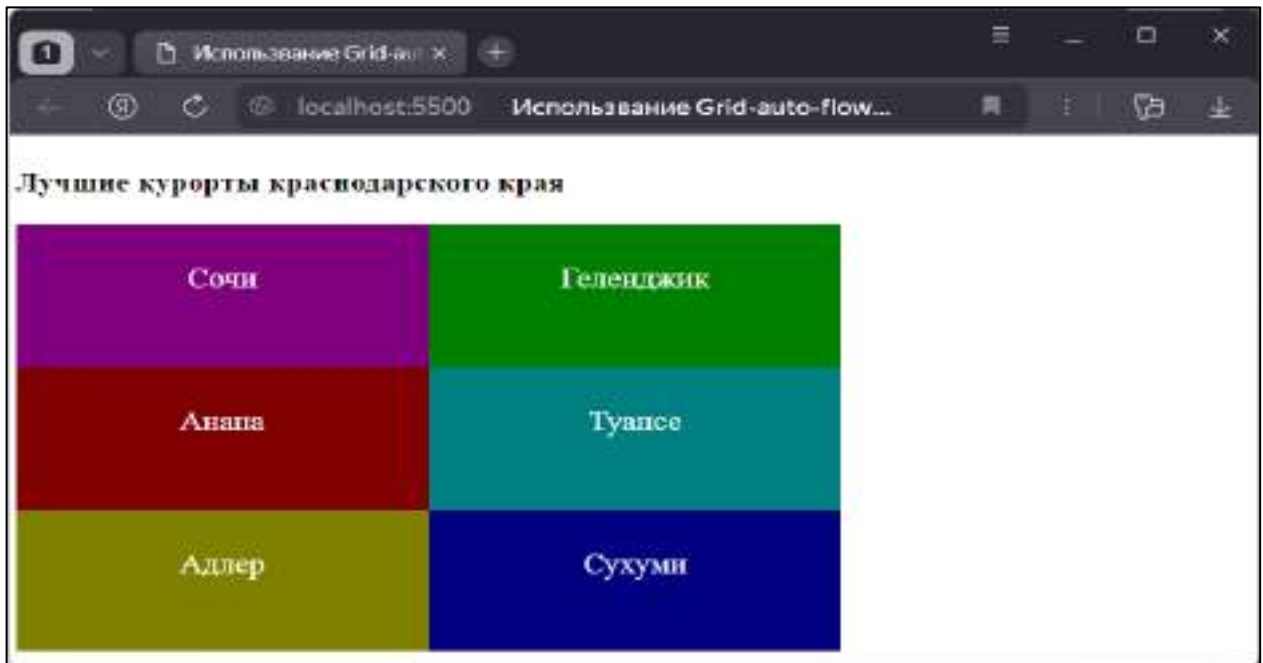


Рисунок 10.15 – Использование `Grid-auto-flow` в `Grid Layout CSS3`

10.7.2 Свойство `order`

Свойство `order` позволяет задать порядок элементов. По умолчанию для каждого элемента в гриде это свойство имеет значение 0. Поэтому элементы располагаются друг за другом, как они определены в разметке `html`. Но можно переопределить этот порядок. В нижеприведенном примере задано изменение порядка с помощью свойства `order`.

```
<head> <style>  
    *{ box-sizing: border-box; }  
    html, body {margin:1; padding:1; }  
    .container { height: 80vh; display: grid;
```

```

        grid-template-columns: repeat(3, 1fr);
        grid-template-rows: repeat(3, 1fr); }

.item { text-align:center;
        font-size: 1.2em; padding: 1.4em;}

.l-item{ order: 1; } .f-item{ order: -1; }

.col1 {background-color: Fuchsia;}
.col2 {background-color: Red;}
.col3 {background-color: Yellow;}
.col4 {background-color: Lime;}
.col5 {background-color: Aqua;}
.col6{background-color: Blue ;} </style> </head>

<body> <h2>Лучшие пляжи города-курорта Сочи</h2>
        <div class="container">
            <div class="item col1">“Ривьера”. Сочи, Ривьерский переулок, 5.</div>
            <div class="item col2 l-item">”Мандарин”. Адлер, ул. Бестужева, 1. </div>
            <div class="item col3 f-item">“Олимпийский пляж”. Поселок Сириус, рядом со стадионом
“Фишт”. </div>
            <div class="item col4">“Пляж Лоо”. Сочи, ул. Лучезарная, 14. </div>
            <div class="item col5">“Альбатрос”. Микрорайон Новый Сочи, ул. Политехническая.</div>
            <div class="item col6"> “Жемчужина”. Сочи, ул. Черноморская, 11.</div> </div> </body>

```

Реализация данного кода представлена на рисунке 10.16.

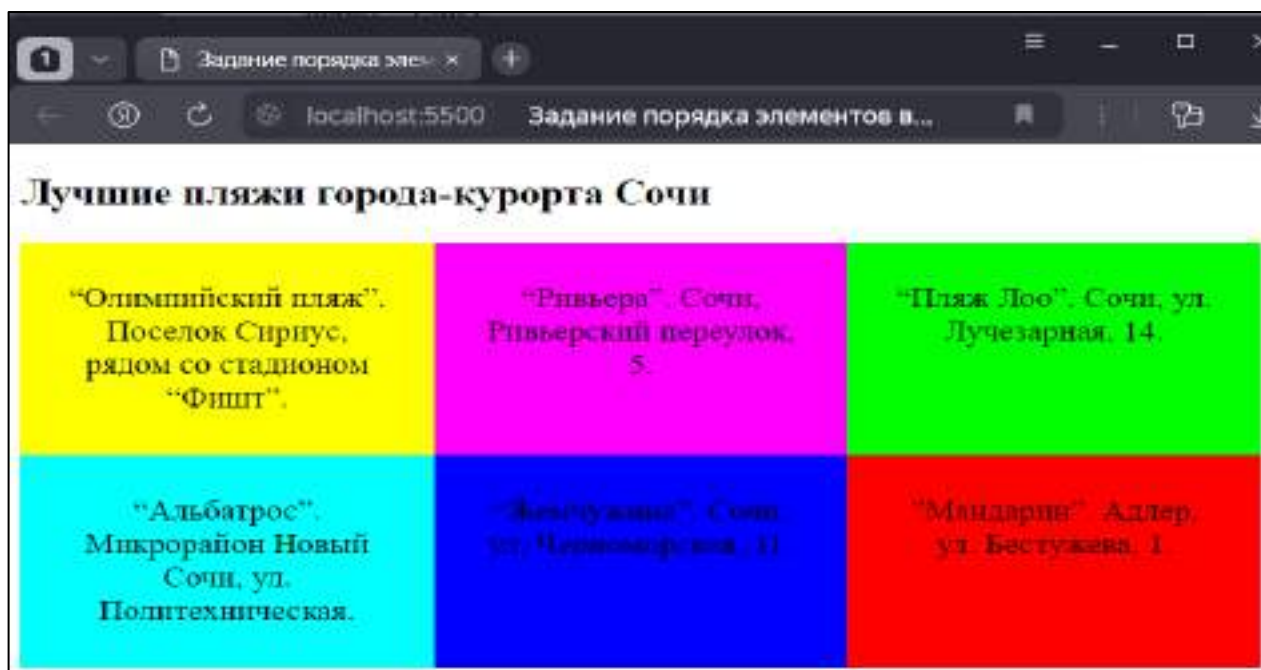


Рисунок 10.16 – Задание порядка элементов в Grid Layout в CSS3

Элементы с классом l-item имеет порядок 1, поэтому они будут располагаться после других элементов, у которых порядок равен 0 или меньше.

Если же необходимо поставить одни элементы перед другими, то можно использовать отрицательное значение для свойства order. Так, у третьего элемента порядок равен -1, что меньше, чем у других элементов.

10.8 Создание макета страницы в Grid Layout

Рассмотрим создание простейшего адаптивного стандартного макета веб-страницы, который состоит из шапки, подвала, основного содержимого, блока навигации и сайдбара. Для этого определяется следующая веб-страница:

```
<head> <style> *{ box-sizing: border-box;
    html, body {margin:1; padding:1;}
    .container {height:50vh; display: grid;
        grid-template-areas: "header" "."
        "snegur" "."        "content" "."        "dedsnow" "."        "footer";
        grid-template-columns: 1fr; grid-template-rows: 80px 5px 100px 5px 1fr 5px 120px 5px 50px;}
        .header {grid-area: header; background-color: LightSkyBlue; }
        .snegur {grid-area: menu; background-color: PowderBlue; }
        .dedsnow {grid-area: sidebar; background-color: PowderBlue; }
        .content {grid-area: content; background-color: LightSteelBlue; }
        .footer {grid-area: footer; background-color: SkyBlue;}
        h3{ text-align:center;}
        @media screen and (min-width: 480px) {
            .container { height:50vh; display: grid;
                grid-template-areas: "header header header header header"
                ". . . ." "snegur. content . dedsnow"
                ". . . ." "footer footer footer footer footer";
                grid-template-columns: 300px 5px 1fr 5px 300px;
                grid-template-rows: 50px 5px 1fr 5px 50px;}} </style>
```

```

</head> <body>
<div class="container"> <div class="header">
<h3>Лучший праздник – Новый год! </h3></div>
<div class="content"><p>Самый волшебный и долгожданный праздник в году для
взрослых и детей – это Новый год! Его ждут с нетерпением в разных странах мира, чтобы
загадать заветные желания и получить подарки о которых мечтали. Главным символом
Нового года считается лесная красавица - елка, которую украшают гирляндами и
разноцветными шарами. В новогоднюю ночь, под бой курантов, на своих северных оленях Дед
Мороз приносит подарки каждому ребенку. Помогает ему творить волшебство, его вечная
спутница - внучка – Снегурочка! </h3>
</div>
<div class="snegur"><h3>Снегурочка – внучка Деда Мороза. Русский сказочный и
новогодний персонаж, внучка Деда Мороза, его постоянная спутница и помощница. В
русском фольклоре Снегурочка является персонажем народной сказки о сделанной из снега
девочке Снегурке, которая ожила. Она обычно носит синюю шубу и корону. Русская
Снегурочка - светловолосая, стройная и очень добрая. </h3>
</div>
<div class="dedsnow"><h3>Дед Мороз - главный сказочный персонаж на русском
празднике Нового года. Прототипами являются Мороз - персонаж славянского сказочного
фольклора и персонажи, основанные на образе святого Николая Чудотворца. Родиной Деда
Мороза является небольшой город в Вологодской области Великий Устюг, который последнее
время называют Сказочной столицей России.</h3>
</div>
<div class="footer"><h3>&copy; Copyright Новый год 2024-2025 год </h3></div>
</div> </body>

```

В данном примере в стилях имеются два разных определения грида. Одно определение грида для мобильных устройств (условно в качестве ширины устройств выбрано значение в 480px) (рисунок 10.17):

```

.container { height: 50vh;display: grid;grid-template-areas: "header" ". "
                "snegur" ". "
                "content" ". "
                "dedsnow" ". "
                "footer";

```

```

grid-template-columns: 1fr; grid-template-rows: 80px 5px 100px 5px 1fr 5px 120px 5px 50px;}

```

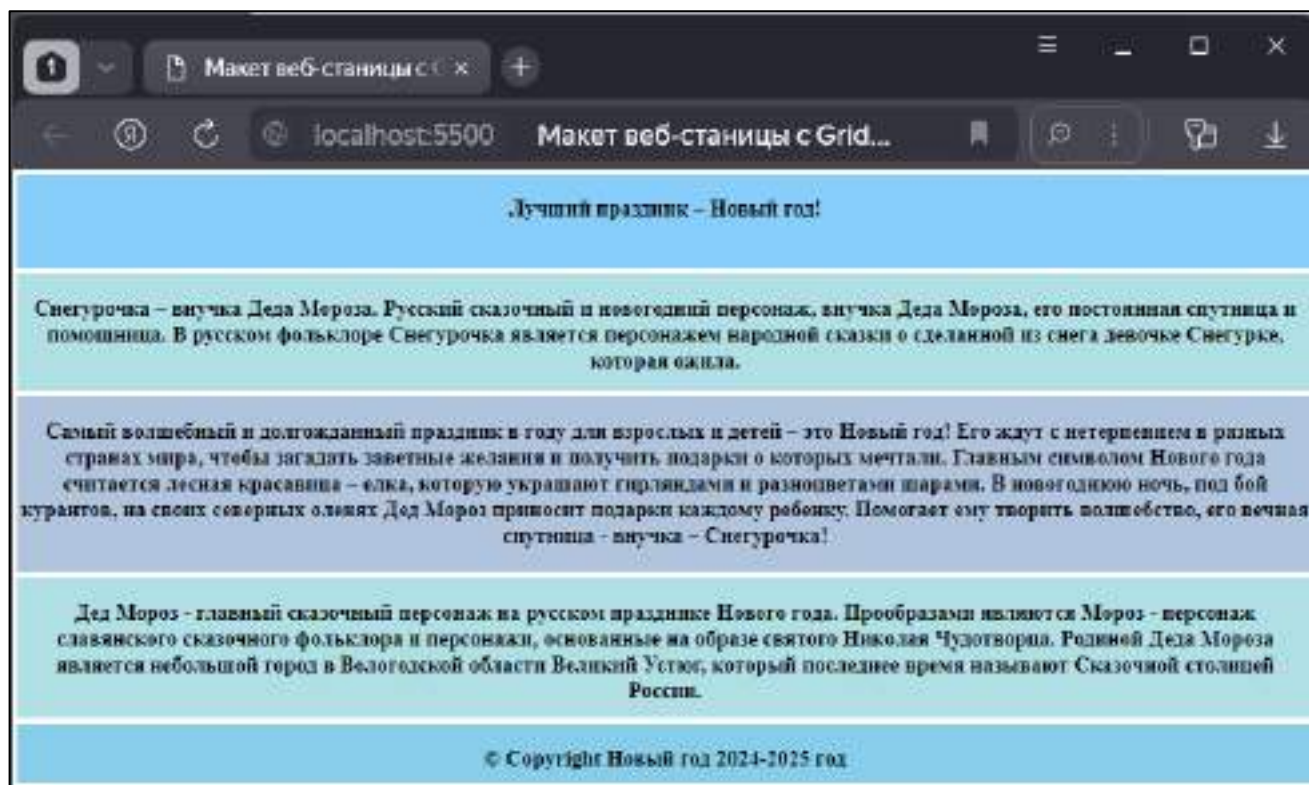



Рисунок 10.17 - Макет веб-станции с Grid Layout в CSS3

В таком состоянии грид имеет только один столбец и 5 строк для каждой области плюс 4 строки-разделители. При увеличении ширины экрана в действие вступает другое определение грида (рисунок 10.18).

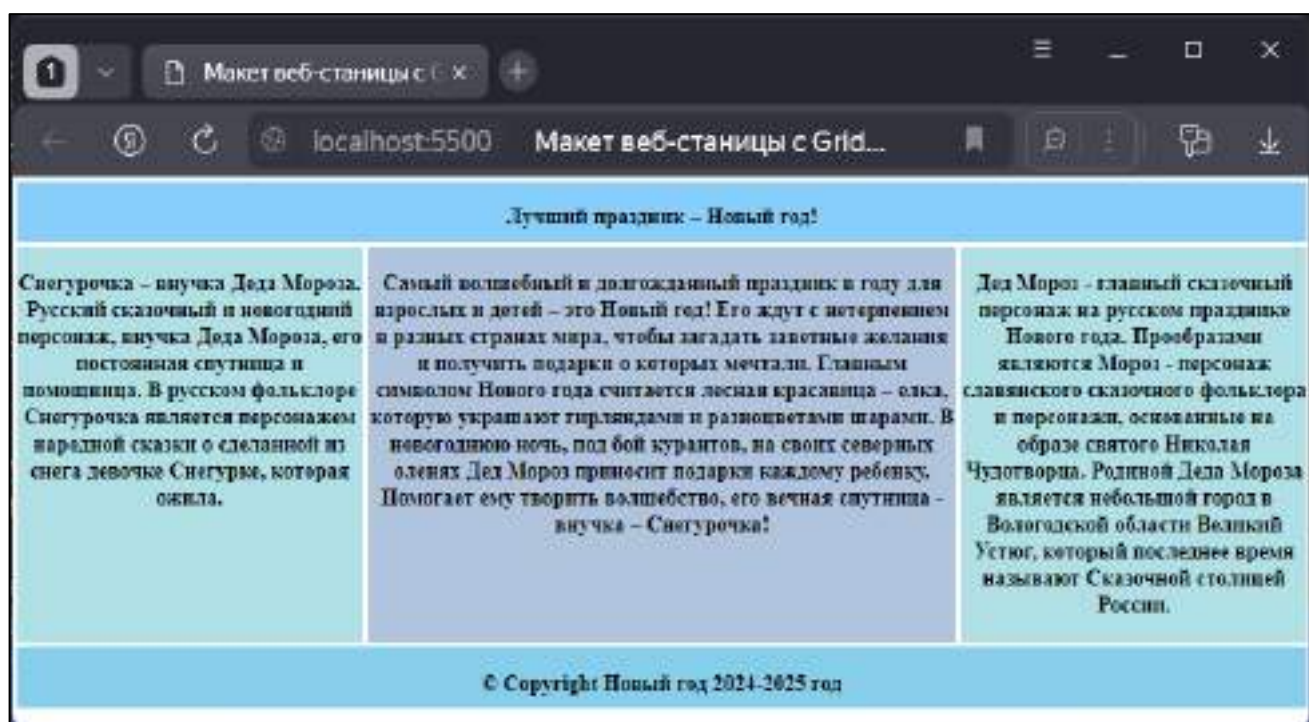


Рисунок 10.18 - Адаптивный макет станции с Grid Layout в CSS3

```
.container {height:50vh; display: grid; grid-template-areas: "header header header header header"  
                ". . . . ." "snegur . content . dedsnow"  
                ". . . . ." "footer footer footer footer footer";  
grid-template-columns: 300px 5px 1fr 5px 300px; grid-template-rows: 50px 5px 1fr 5px 50px;}
```

В этом случае столбцов и строк будет по пять.

10.9 Контрольные вопросы

- 1) Дать определение Grid Layout, Grid Container.
- 2) Рассказать о позиционировании элементов.
- 3) Рассказать о направлении и порядке элементов в grid контейнере.
- 4) Рассказать об области применения грида.
- 5) Рассказать о создании макета страницы с помощью Grid Layout.

10.10 Тестирование по главе

- 1) Для создания grid-контейнера указывается свойство ...
 - а) display: inline-grid;
 - б) display: grid-inline;
 - в) display: flex-grid;
 - г) display: block-grid.
- 2) Укажите, верно, записанное свойство grid
 - а) grid: grid-template-columns/grid-template-rows;
 - б) grid: grid-template-rows/grid-template-columns;
 - в) grid: [grid-template-columns] [grid-template-rows];
 - г) grid: [grid-template-rows] [grid-template-columns].

3) Укажите правильный вариант сокращения строки `grid-template-rows: 5em 5em 5em 5em`

- а) `grid-template-rows: 4 * 5em;`
- б) `grid-template-rows: 5em * 4;`
- в) `grid-template-rows: repeat (5em, 4);`
- г) `grid-template-rows: repeat (4, 5em).`

4) Для установки пропорциональных размеров в `grid` применяется единица измерения ...

- а) `pt;`
- б) `rem;`
- в) `fr;`
- г) `px.`

5) В `grid` указаны столбцы: `2fr 25px 1fr`, ширина `grid` равна `100px`. Вычислите ширину первого столбца

- а) `50px;`
- б) `37.5px;`
- в) `75px;`
- г) `25px.`

6) По умолчанию каждый элемент в `grid` позиционируется ...

- а) в одну ячейку по порядку;
- б) в несколько ячеек по порядку;
- в) с нижнего левого угла;
- г) с правого верхнего угла.

7) Укажите правильное определение свойства `grid-column`

- а) `grid-column: [grid-column-start] [grid-column-end];`

- б) `grid-column: grid-column-start / grid-column-end;`
- в) `grid-column: [grid-column-end] [grid-column-start];`
- г) `grid-column: grid-column-end / grid-column-start.`

8) Укажите правильное определение свойства `grid-area`

- а) `grid-area: row-start / column-start / row-end / column-end;`
- б) `grid-area: row-start / row-end / column-start / column-end;`
- в) `grid-area: column-start / row-start / column-end / row-end;`
- г) `grid-area: column-start / column-end / row-start / row-end.`

9) Свойство ... позволяет изменить направление элементов

- а) `grid-auto-flow;`
- б) `grid-template;`
- в) `grid-area;`
- г) `grid.`

10) Как будут располагаться ячейки при указании свойства `grid-template-areas:` “1 1” “2 3” “2 4”? В ответе указано расположение слева направо сверху вниз, пробелом разделены строки `grid`

- а) 122 134;
- б) 11 23 24;
- в) 11 32 42;
- г) 24 23 11.

11 Использование переменных в CSS

11.1 Стилизация с помощью переменных

Как и в языках программирования, в CSS можно определять переменные. Переменные в CSS могут хранить стандартные значения, которые можно присвоить, обычным свойствам CSS, например, цвет фона, цвет шрифта, высоту шрифта, ширину и высоту элементов и так далее. Затем их можно многократно использовать в различных частях определения стилей [7].

Хотя используется термин “переменные” (css variables), официально они называются custom properties (кастомные или настраиваемые свойства).

Определение переменных CSS начинается с префикса --, например, --my-color. Они могут быть определены для любого элемента. Далее приведен пример определения переменных для задания цвета текста, начертания семейства шрифтов, а также размера текста.

```
<head> <style> div {
    --text-color1:MediumSeaGreen;
    --text-color2: LightSkyBlue; --text-color3: HotPink;
    --text-size1: 30px; --text-size2: 20px;
    --text-size3: 15px; --text-font-family-verdana: Verdana;
    --text-font-family-fantasy: fantasy;
    --text-font-family-courier-new: 'Courier New';}
#div1 {font-size: var(--text-size1);  var(--text-color1);
    font-family: var(--text-font-family-verdana), serif;}
#div2 {font-size: var(--text-size2); color: var(--text-color2);
    font-family: var(--text-font-family-fantasy), serif;}
#p1 {font-size: var(--text-size3); var(--text-color3);
    font-family: var(--text-font-family-courier-new);} </style></head>
<body> <div id="div1">Волшебник изумрудного города</div>
```

`<div id="div2">Сказка о приключениях маленькой девочки Элли и её собачки Тотошки. Однажды во время грозы, Элли попала в чудесную страну, а чтобы вернуться назад, ей нужно было помочь трем существам исполнить заветные желания. </div>`

`<div><p id="p1">Сказка была написана Александра Волковым в 1939 году на основе сюжета сказки американского писателя Лаймена Фрэнка Баума “Удивительный волшебник из страны Оз”.</p></div></body>`

Реализация данного кода представлена на рисунке 11.1.

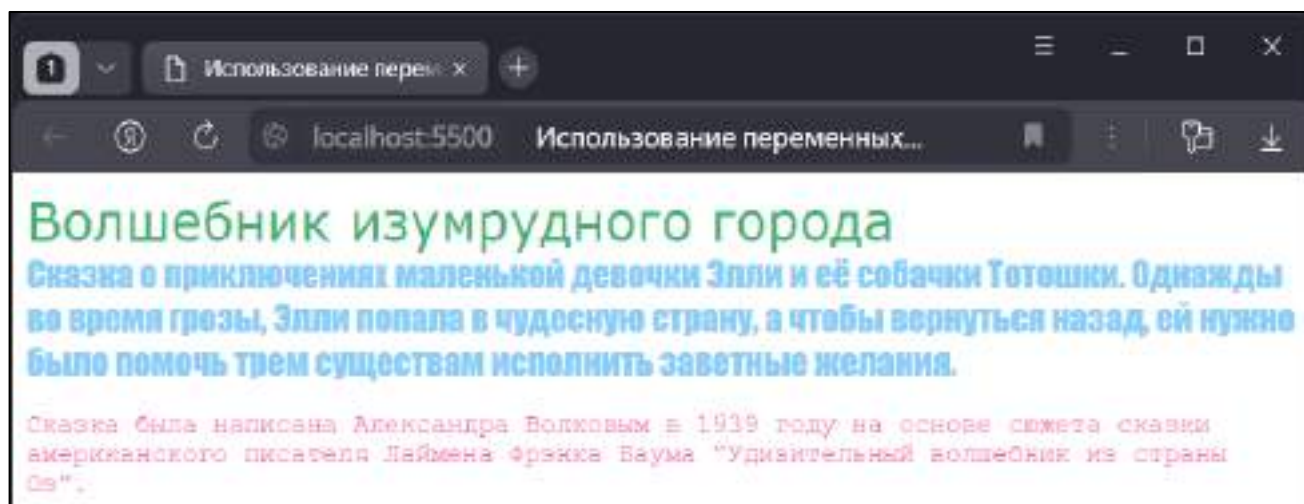


Рисунок 11.1 - Использование переменных в CSS3

Здесь в коде CSS для элемента `div` определены несколько переменных:

```
--text-color1:MediumSeaGreen;
--text-color2: LightSkyBlue;
--text-color3: HotPink;
--text-size1: 30px;
--text-size2: 20px;
--text-size3: 15px;
--text-font-family-verdana: Verdana;
--text-font-family-fantasy: fantasy;
--text-font-family-courier-new: 'Courier New';}
```

С помощью выражения `var()` можно ссылаться на эти переменные в любой части кода CSS:

```
#div1 {font-size: var(--text-size1);
      var(--text-color1); font-family:
      var(--text-font-family-verdana), serif;}
```

```
#div2 {font-size: var(--text-size2); color: var(--text-color2);
      font-family: var(--text-font-family-fantasy), serif;}
#p1 {font-size: var(--text-size3); var(--text-color3);
     font-family: var(--text-font-family-courier-new);}
```

Если потребуется изменить какие-то параметры текста (цвет, высоту, гарнитуру), достаточно будет изменить значение переменной.

Подобные переменные можно определить для любого элемента. При этом они наследуются дочерними элементами.

Если же необходимо, чтобы переменные могли бы использоваться глобально для всех элементов, тогда их определяют для элемента: `root`

11.2 Создание тем CSS с помощью переменных

Использование переменных в CSS открывает возможность создания и применения тем на веб-странице. Например, определяется следующая веб-страница [14]:

```
<head> <style>
:root { --panel-bg-color: #ebebeb;
        --container-bg-color: white;
        --text-color: black;}
:root[theme="dark"] {
        --panel-bg-color: #646464;
        --container-bg-color: black;
        --text-color: white;}
* {margin: 0;}
html { height: 100%;}
body {height: 100%; font-family: Verdana; display: flex; flex-direction: column; color: var(--text-color); }
.navbar { align-items: center;
         display: flex; justify-content: space-between; padding: 10px;
```

```

background: var(--panel-bg-color); }
.navbar a {padding: 10px; cursor: pointer;}
.container {flex: 1; padding: 10px;
background: var(--container-bg-color); }
footer { padding: 10px;
background: var(--panel-bg-color); }input[type="button"] {
color: var(--text-color);
background: var(--panel-bg-color);cursor: pointer;
padding: 0.3 rem; margin: 0.2 rem;
font-size: 1rem; } </style> </head>
<body> <nav class="navbar">
<div><a>Введение в Kotlin</a>
<a>IDE IntelliJ Idea</a>
<a>IDE Android Studio</a></div>
<input type="button" value="Сменить тему" id="toggle-theme" />
</nav> <div class="container">
<h2 class="title">Разработка мобильных приложений с помощью языка
программирования Kotlin</h2>
<p class="content">Kotlin - это язык разработки общего назначения, используемый
в основном для разработки мобильных приложений Android. Помимо приложений Android,
Kotlin также полезен для следующих целей: разработка на стороне сервера. Разработка веб-
приложений на стороне сервера традиционно использует Java. </p> </div><footer>
<p><p>&copy; Copyright 2024</p> </footer>
<script>
const toggleBtn = document.querySelector("#toggle-theme");
toggleBtn.addEventListener("click", function () {
if (document.documentElement.hasAttribute("theme")) {
document.documentElement.removeAttribute("theme");}
else {
document.documentElement.setAttribute("theme", "dark"); } });
</script> </body>
В стилях страницы определены две темы, которые содержат три переменных:
:root { --panel-bg-color: #ebebeb;
--container-bg-color: white;

```



```

--text-color: black;}
:root[theme="dark"]
{ --panel-bg-color: #646464;
  --container-bg-color: black;
  --text-color: white;}

```

Первая тема - условно светлая, вторая - условно темная. При темной теме корневой элемент, то есть элемент `<html>` будет иметь атрибут `theme="dark"`. Затем эти переменные используются для установки стилевых свойств отдельных элементов.

Для переключения тем у кнопки через несложный код javascript установлен обработчик нажатия, который проверяет наличие атрибута `"theme"` (что будет означать, что установлена темная схема). И при наличии атрибута убирает его, а при его отсутствии, наоборот, устанавливает.

```

const toggleBtn = document.querySelector("#toggle-theme");
toggleBtn.addEventListener("click", function() {
  if(document.documentElement.hasAttribute("theme")) {
    document.documentElement.removeAttribute("theme");}
  else
    {document.documentElement.setAttribute("theme", "dark"); }
});

```

В результате при нажатии на кнопку произойдет глобальное переключение стилей веб-страницы. Реализация данного кода представлена на рисунках 11.2-11.3.

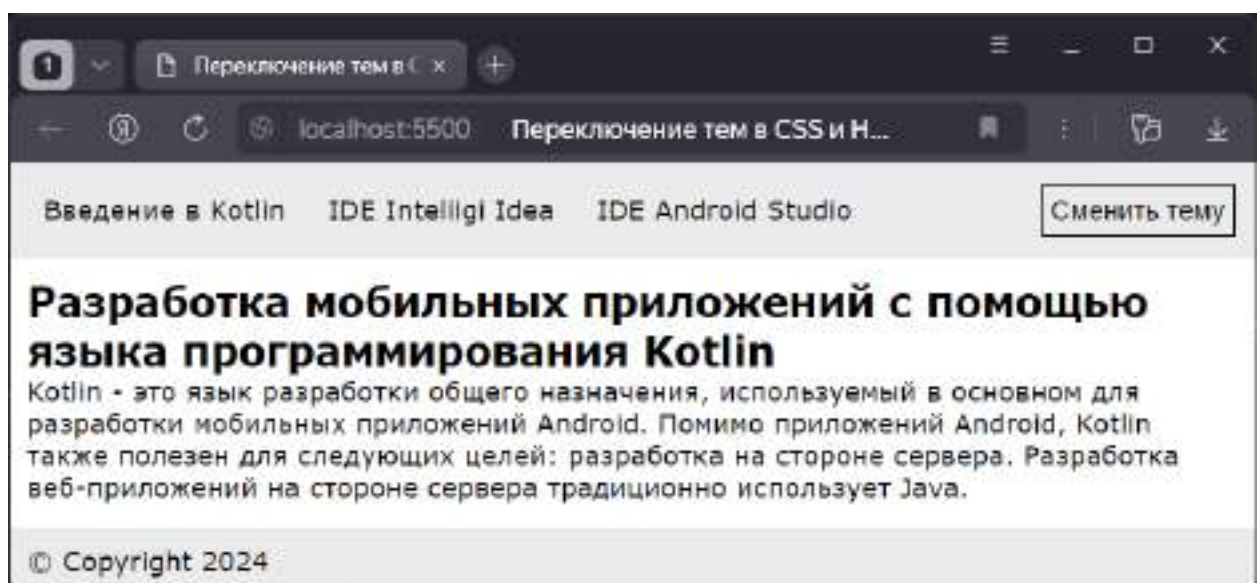


Рисунок 11.2 - Переключение тем в CSS и HTML (светлая тема)

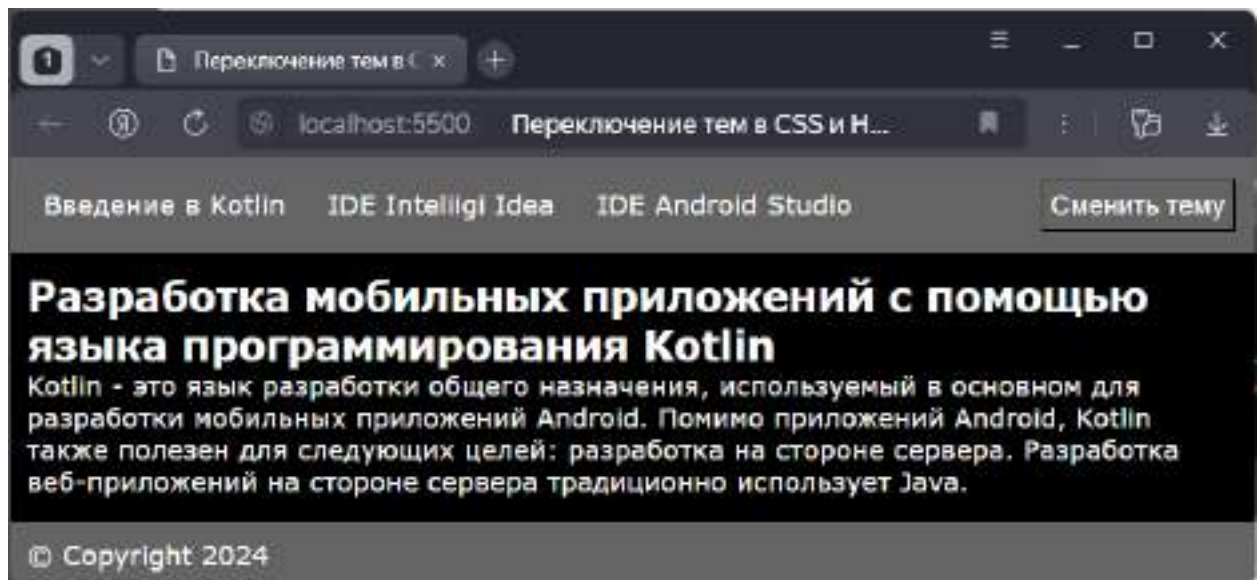


Рисунок 11.2 - Переключение тем в CSS и HTML (темная тема)

11.3 Стили CSS как хранилище данных

Применение переменных в CSS не ограничивается настройкой стилизации. Их предназначение более широко - а именно хранение состояния. Благодаря этому даже можно определить с помощью стилей CSS и переменных своеобразную базу данных или хранилище данных, которое может в определенных сценариях применяться для хранения данных веб-страницы.

Например, определяется файл стилей, который называется `user.css` со следующим содержимым:

```
.user { --name: "Дмитрий";  
        --age: "25";  
        --email: "DmitrySergeev@mail.com";  
        --address: "г. Оренбург, улица Просторная, д.10, кв. 15";}
```

Здесь для класса `user` определены четыре переменных, которые хранят некоторые значения. В данном случае все значения представляют строки.

Теперь определяется следующая html-страница, на которой будет подключаться вышеопределенный файл стилей.

```
<head> <link rel="stylesheet" type="text/css" href="user.css"/>
  <style> .user-name:after { content: var(--name); }
        .user-age:after { content: var(--age); }
        .user-email:after { content: var(--email); }
        .user-address:after { content: var(--address); } </style> </head> <body>
<div class="user">
<h2 class="user-name">User </h2>
<p class="user-age">Age: </p>
<p class="user-email">Email: </p>
<p class="user-address">Address: </p> </div> </body>
```

Реализация данного кода представлена на рисунке 11.3.

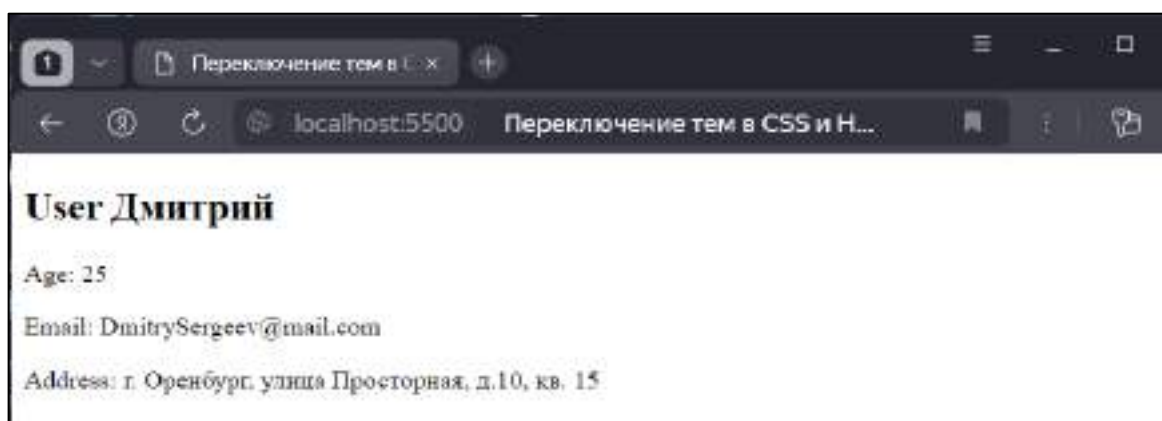


Рисунок 11.3 – Использование CSS как хранилище данных

Для вывода данных на страницу применяется элемент `<div class="user">`, в котором определены четыре html-элемента: один заголовок `h2` и три параграфа. И для каждого элемента определен свой класс.

В элементе `<style>` определяются стили для этих элементов, которые применяют переменные из подключаемого файла `user.css`. Все эти стили однотипны, они устанавливают текст элемента, который добавляется после уже имеющегося текстового содержимого. Например:

```
.user-name:after { content: var(--name); }
```

::after создает псевдоэлемент в конце html-элемента. И у этого псевдоэлемента в качестве содержимого устанавливается текст из переменной --name.

Стили для остальных элементов идентичны. Таким образом, ожидается, что значения, которые хранятся в файле user.css, будут выводиться на веб-страницу. Кинем страницу в браузер и на ней будут данные из подключённого файла css.

Хранение текстовых данных и вывод их в текстовые блоки на веб-страницу, естественно, это только частный случай. В более широком смысле переменные CSS позволяют хранить некоторое состояние, которое применяется к html-странице.

11.4 Контрольные вопросы

- 1) Использование переменных в CSS. Описать направления использования.
- 2) Описать стилизацию страниц с помощью переменных CSS.
- 3) Создание различных тем в CSS с помощью переменных.
- 4) Привести примеры создания темной и светлой тем в CSS.
- 5) Использовании стилей CSS как хранилище данных. Привести примеры создания хранилища данных.

11.5 Тестирование по главе

- 1) Переменные в CSS могут хранить ...
 - а) теги;
 - б) атрибуты;
 - в) свойства;
 - г) значения свойств.

- 2) Определение переменных начинается с ...
- а) “&;
 - б) “- -”;
 - в) “\$”;
 - г) “@”.
- 3) С помощью какого выражения можно ссылаться на переменные?
- а) let;
 - б) var;
 - в) variables;
 - г) properties.
- 4) Как использовать переменные глобально для всех элементов?
- а) * {...};
 - б) :all {...};
 - в) :root {...};
 - г) {...}.
- 5) Официальное название переменных в CSS - ...
- а) кастомные свойства;
 - б) переменные;
 - в) настраиваемые переменные;
 - г) css-переменные.
- 6) Как можно избежать ошибок при определении переменных?
- а) Задать резервное значение первым параметром в var;
 - б) Задать резервное значение вторым параметром в var;
 - в) Указать резервное значение после указания переменной;
 - г) Указать резервное значение до указания переменной.

7) Что, помимо стилизации, могут хранить переменные в CSS?

- а) таблицы;
- б) файлы;
- в) состояния;
- г) код CSS/HTML.

8) Укажите правильное задание переменной в свойстве

- а) color: --col;
- б) color: var (col);
- в) color: var (--col);
- г) color: (--col)var;

9) Укажите правильное задание резервного значения переменной

- а) color: var (--col_1, var (--col_2));
- б) color: var(col_1) var (col_2);
- в) color: (var (col_1), var(col_2));
- г) color: (var(--col_1), var(--col_2)).

10) Укажите правильное задание присвоения текстовых значений для последующего вывода их на страницу

- а) .user {content: var(--name);};
- б) .user:after {content:var (--name);};
- в) .user {content: var(name);};
- г) .user:after {content:var (name);}.

Заключение

HTML5 (HyperText Markup Language) - стандартизированный язык разметки гипертекста для документов, предназначенный для отображения веб-страниц в браузере. Веб-браузеры получают HTML-документ с сервера по протоколам HTTP/HTTPS или открывают его с локального диска, затем интерпретируют код в интерфейс, который будет отображаться на экране монитора.

CSS (каскадные таблицы стилей) - формальный язык оформления и описания внешнего вида веб-страницы, написанный с использованием языка разметки.

Пособие предназначено для обучения студентов современным средствам и подходам к разработке клиентской части сайтов, приобретение навыков разработки и отладки веб-приложений с использованием языка разметки HTML5 и каскадных таблиц стилей CSS3.

После окончания изучения представленного учебного пособия студенты будут уметь самостоятельно разрабатывать веб-приложения на стороне клиента, используя элементы форматирования языка HTML5 и стилизацию с помощью каскадных таблиц стилей CSS3.

Список использованных источников

- 1 Попов Е. Руководство по HTML5 и CSS3. Режим доступа: <https://metanit.com/web/html5/>.
- 2 Дубовик Е.В. Справочник HTML. Кратко, быстро, под рукой. М.: Наука и техника. 2023. - 288 с.
- 3 Трепачев Д. Учебник по верстке для новичков. Режим доступа: <https://code.mu/ru/markup/book/prime/>
- 4 Чиртик А.А. HTML: Популярный самоучитель. Режим доступа: <https://mybook.ru/author/aleksandr-anatolevich-chirtik/html-populyarnyj-samouchitel/read/>.
- 5 Кириченко А.В. Web на практике. CSS, HTML, JavaScript, MySQL, PHP для fullstack-разработчиков / А.В. Кириченко, А.П. Никольский, Е.В. Дубовик - СПб.: НАУКА и ТЕХНИКА, 2021. - 432 с.
- 6 Эберт Елена: Шпаргалки для начинающего верстальщика HTML/CSS / Эберт Елена. - Изд. Сист. Ridero, 2021. - 103 с.
- 7 Купер Нейт: Как создать сайт. Комикс-путеводитель по HTML, CSS и WordPress. / Купер Нейт. - М.: Манн, Иванов и Фербер, 2019. - 266 с.
- 8 Грант Кит: CSS для профи/Кит Грант. Режим доступа: <https://codelibrary.info/online?url=L2Rvd25sb2FkLzE0NDhfY3NzLWRseWEtcHJvZmkuсGRm>
- 9 Кириченко А.В. HTML5 + CSS3. Основы современного WEB-дизайна. Режим доступа: <https://codelibrary.info/online?url=L2Rvd25sb2FkLzE0MTNfaHRtbDUtY3NzMy5wZGY=>
- 10 Сидельников Грег: Наглядный CSS. Режим доступа: <https://codelibrary.info/online?url=L2Rvd25sb2FkLzE3MzhfTmFnbHlhZG55aXktQ1NTLnBkZg==>

11 Дронов В. HTML и CSS. 25 уроков для начинающих / В. Дронов. - БХВ-Петербург, 2020. - 381 с.

12 Дакетт Джон: HTML и CSS. Разработка и дизайн веб-сайтов. / Джон Дакетт. - Эксмо, 2019. - 480 с.

13 Самарев Р.С. Создание простейших HTML-страниц, валидаторы кода. Каскадные таблицы стилей CSS / Р. С. Самарев, К. В. Кучеров. - МГТУ им. Н.Э. Баумана (национальный исследовательский университет), 2021. - 64 с.

14 Никсон Робин: Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript, CSS и HTML5. 6-е изд. / Никсон Робин. - Питер, 2023. - 832 с.

15 Диков А. В. Клиентские технологии веб дизайна. HTML5 и CSS3: учебное пособие / А. В. Диков. - Санкт-Петербург: Лань, 2019. - 188 с.

16 Фрэйз Бен: Отзывчивый дизайн на HTML5 и CSS3 для любых устройств. 3-е изд. - СПб.:Питер, 2022. - 336 с.: ил.

17 Кириченко А. В. Web на практике. CSS, HTML, JAVASCRIPT, MySQL, PHP для fullstack-разработчиков. Режим доступа: <https://codelibrary.info/online?url=L2Rvd25sb2FkLzE0NTBfV2ViX25hX3ByYWt0aWtlLi1DU1MtSFRNTC1KYXZhU2NyaXB0LU15U1FMLnBkZg==>

18 Роббинс Д. Н. Веб-дизайн для начинающих. HTML, CSS, JavaScript и веб-графика. Режим доступа: <https://codelibrary.info/online?url=L2Rvd25sb2FkLzExMDVfV2ViLWRpemF5bi1kbHlhLW5hY2hpbmF5dXNoY2hpaC1IVE1MLUNTUy1KYXZhU2NyaXB0LWktdmViLWdyYWZpa2EucGRm>

19 Эберт Е. Шпаргалки для начинающего верстальщика HTML/CSS. Режим доступа: <https://codelibrary.info/online?url=L2Rvd25sb2FkLzE2MDlfU2hwYXJnYWxrYS1IVE1MLUNTUy5wZGY=>