Руководство для начинающих по HTML и CSS

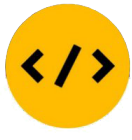
Пошаговое руководство с примерами и упражнениями

Кевин Уилсон

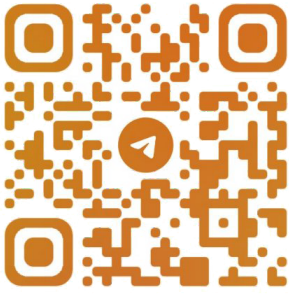
Apress®

Руководство для начинающих по HTML и CSS

Пошаговое руководство с примерами и упражнениями



Кевин Уилсон



Apress®

Руководство для начинающих по HTML и CSS: пошаговое руководство с примерами и упражнениями

Кевин Уилсон
УИДНС, Великобритания

ISBN-13 (бумажная): 978-1-4842-9249-5 ISBN-13 (элктронная): 978-1-4842-9250-1
https://doi.org/10.1007/978-1-4842-9250-1

Управляющий директор Apress Media LLC: Велмоед Спар
Редактор отдела закупок: Джеймс Робинсон
Редактор отдела разработки: Джеймс Маркхэм
Координирующий редактор: Гриффин Винклер
Обложка разработана eStudioCalamar

Изображение обложки разработано векторным соком на Freepik

Распространяется в книжной торговле по всему миру компанией Springer Science+Business Media New York, 1 New York Plaza, Suite 4600, New York, NY 10004-1562, США. Телефон 1-800-SPRINGER, факс (201) 348-4505, электронная почта order-ny@springer-sbm.com или посетите сайт www.springeronline.com. Apress Media, LLC является калифорнийской компанией LLC, а единственным участником (владельцем) является Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc — корпорация штата **Делавэр**.

Для получения информации о переводах отправьте электронное письмо по адресу booktranslations@springernature.com; Чтобы получить права на перепечатку, мягкую обложку или аудио, отправьте электронное письмо по адресу bookpermissions@springernature.com.

Издания Apress можно приобрести оптом для академического, корпоративного или рекламного использования. Версии электронных книг и лицензии также доступны для большинства изданий. Для получения дополнительной информации посетите нашу веб-страницу оптовых продаж печатных и электронных книг по адресу http://www.apress.com/bulk-sales.

Любой исходный код или другие дополнительные материалы, на которые ссылается автор в этой книге, доступны читателям на GitHub по адресу https://github.com/Apress.

Напечатано на бескислотной бумаге

Об авторе

Имея более чем 20-летний опыт работы в компьютерной индустрии, **Кевин Уилсон** сделал карьеру в сфере технологий и обучения их использованию. Получив степень магистра в области компьютерных наук, разработки программного обеспечения и мультимедийных систем, Кевин занимал различные должности в ИТ-индустрии, включая графический и веб-дизайн, цифровое кино и фотографию, программирование и разработку программного обеспечения, разработку и управление корпоративными сетями, создание компьютерных систем. и ИТ-поддержка. В настоящее время он преподает информатику в колледже и работает тренером по информационным технологиям в Англии, одновременно работая над докторской диссертацией.

О техническом рецензенте

Джонатон Симпсон — владелец продукта и инженер, живущий в Великобритании. Он окончил UCL в 2015 году. Имея многолетний опыт работы, он разработал и реализовал множество успешных проектов как самостоятельно, так и в крупных компаниях. Он ведет популярный блог по разработке программного обеспечения под названием fjolt.com и регулярно публикует информационные бюллетени о последних событиях и тенденциях в области JavaScript и веб-разработки.

Введение

Цель этой книги — предоставить вашему вниманию базовый курс по использованию HTML и CSS.

Он представляет собой основу для тех, кто хочет разрабатывать свои собственные веб-сайты, а поскольку книга предназначена для начинающих, она позволит новичку освоиться с основами кодирования на HTML и CSS.

Поскольку это базовый курс, не предполагается наличие предыдущего опыта компьютерного программирования.

На протяжении всей книги мы будем изучать HTML и CSS с проработанными примерами и упражнениями, которые вы сможете выполнить самостоятельно. Мы также представим JavaScript и то, как его можно использовать для добавления интерактивности на веб-сайт, а также с использованием систем управления контентом, таких как WordPress.

Для этой цели мы включили весь исходный код книги в следующий репозиторий: https://github.com/Apress/The-Absolute-Beginner-s-Guide-to-HTML-and-CSS

ГЛАВА 1

Итак, начнем!

Первоначально разработанный в начале 1990-х годов Тимом Бернерсом-Ли, HTML означает язык гипертекстовой разметки и представляет собой язык, используемый для компоновки и форматирования документов Всемирной паутины, предназначенных для отображения в веб-браузере. Другими словами, HTML-код описывает структуру веб-страницы. HTML можно использовать с другими технологиями, такими как каскадные таблицы стилей (CSS) для стилизации и форматирования документа, а также с языками сценариев, такими как JavaScript, для обеспечения функциональности и интерактивности элементов.

Базовые знания HTML необходимы студентам и всем, кто занимается веб-разработкой. Это поможет вам:

- Понимать Всемирную паутину.
- Создавать и настраивать свои собственные веб-сайты. Если вы знаете HTML, вы сможете создать веб-сайт или настроить существующий веб-шаблон.
- Стать веб-разработчиком. Если вы хотите начать карьеру профессионального веб-разработчика, вам необходимы навыки HTML и CSS.

Связь страниц друг с другом

Все веб-страницы связаны между собой посредством интерактивного текста или изображений, называемых гиперссылками. Это называется гипертекстом и позволяет создавать на веб-сайте несколько страниц, которые позволяют

пользователю просматривать страницы, щелкая эти гиперссылки.

Гиперссылки также могут ссылаться на страницы и ресурсы, размещенные на других веб-сайтах.

Ссылки могут быть встроены в текст абзаца в виде подчеркнутого слова, изображения или значка. Это называется гипертекстом.

Как вы можете видеть на Рис. 1-1, слева гиперссылки отображаются в виде голубого текста. При нажатии на ссылку браузер перенаправит вас на связанную страницу.

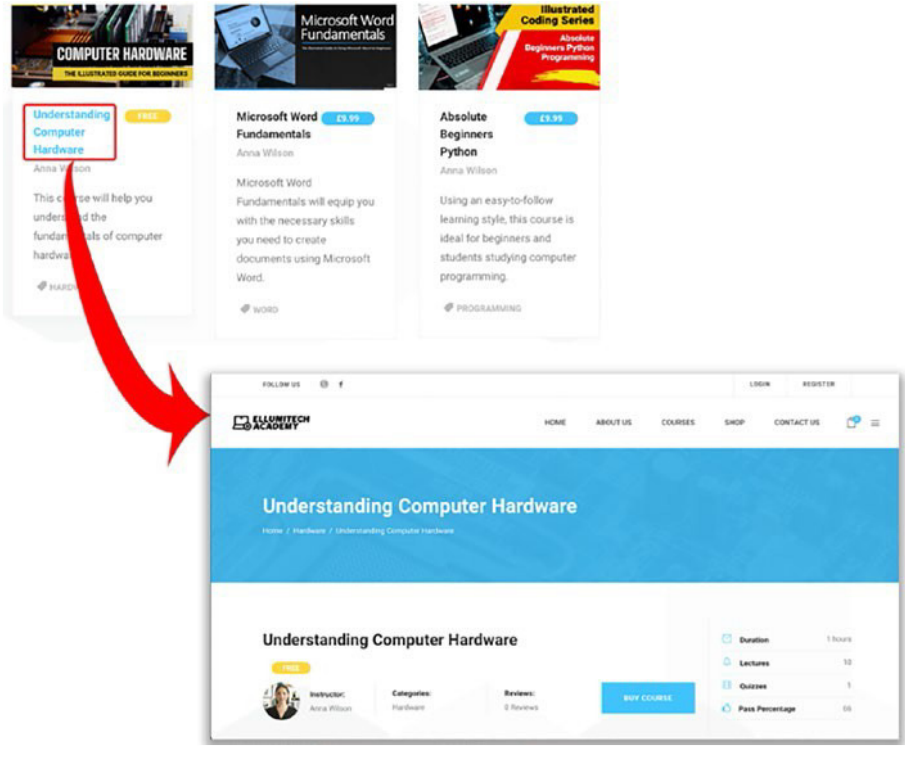


Рис. 1-1. *Гипертекстовый документ*

Где хранятся веб-страницы?

Веб-сайт и его страницы хранятся или размещаются на веб-сервере. Веб-серверы — это компьютеры, обычно работающие под управлением Windows Server или, что чаще, какой-либо разновидности операционной системы Linux, например CentOS. На этих компьютерах работает часть программного обеспечения, называемая веб-сервером. Обычно это Apache, IIS или NGINX (произносится как «Энджин Икс»).

Что такое URL-адрес?

Каждый веб-сайт во Всемирной паутине имеет адрес, называемый URL-адресом (Рис. 1-2) или унифицированным указателем ресурсов.

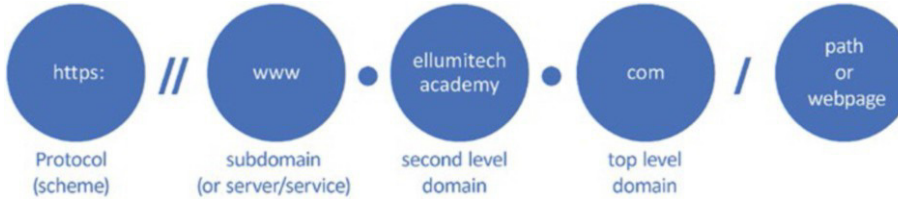


Рис. 1-2. *Анатомия URL*

Сам URL-адрес можно разбить на основные элементы. Давайте подробнее рассмотрим пример:

https://www.ellumitechacademy.com

Часть «www.ellumitechacademy.com» является именем хоста.

Давайте разобьем URL-адрес на несколько частей.

https:// означает защищенный протокол передачи гипертекста, который является протоколом, используемым веб-браузером для подключения к серверу. Это называется схемой.

Вы можете обнаружить другие схемы, такие как ftp://, если вы

подключаетесь к FTP-сайту.

www — это имя сервера, на котором размещена служба или субдомен, в данном случае www для Всемирной паутины, и и которое обычно указывает на ваш каталог public_ html или htdocs на веб-сервере.

Это также может быть и другая служба, например сервер электронной почты:

mail.ellumitechacademy.com

или, возможно, поддомен для интернет-магазина сайта:

shop.ellumitechacademy.com

ellumitechacademy — это доменное имя или название организации, уникальное для этой организации. Это так называемый домен второго уровня.

.com — тип сайта. Это может быть .co.x для компаний конкретной страны (например, .co.uk), .org для некоммерческих организаций или .gov для государственных организаций. Типы известны как доменные имена верхнего уровня и предназначены для идентификации типов компаний, представленных в сети Интернет.

После имени домена вы можете увидеть косую черту, а затем другое имя. Это подкаталог или путь. Например, чтобы получить доступ к каталогу курсов на веб-сервере, мы будем использовать

www.ellumitechacademy.com**/courses**

или для каталога html внутри курсов мы будем использовать

www.ellumitechacademy.com**/courses/html**

Если мы хотим получить доступ к веб-странице или файлу для загрузки, мы добавляем путь и имя файла или документа:

www.ellumitechacademy.com/aboutus.html

или файл в каталоге загрузок:

www.ellumitechacademy.com/downloads/menu.pdf

Конечно, эти файлы и каталоги должны существовать на веб-сервере в каталоге public_html или htdocs (Рис. 1-3). Здесь мы можем увидеть каталоги курсов и загрузок, находящихся на сервере.

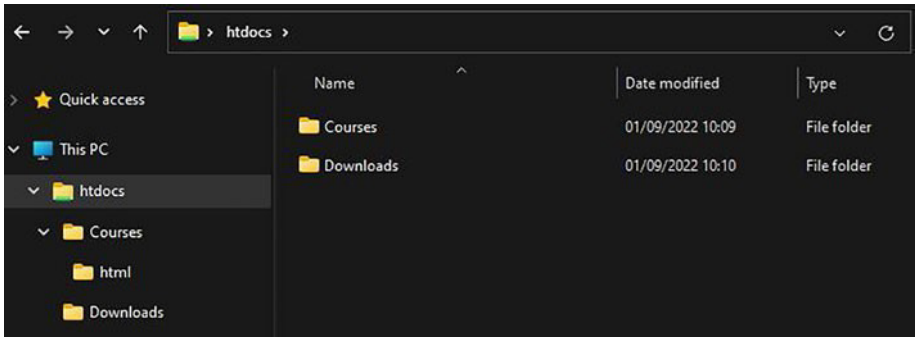


Рис. 1-3. *Структура каталогов на веб-сервере*

Индексные страницы

Веб-сайты создаются внутри каталогов на веб-сервере. Индексный файл — это страница, отображаемая по умолчанию, если не указана другая страница, когда посетитель вводит URL-адрес в свой веб-браузер. Этот индексный файл может быть index.html, index.php или index.py в зависимости от того, какой язык вы используете для разработки своего сайта. Сейчас мы будем использовать index.html.

В нашем примере у нас есть структура каталогов на нашем веб-сервере (Рис. 1-4).

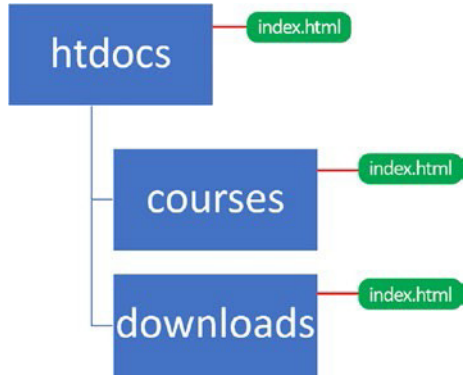


Рис. 1-4. *Структура каталогов*

Внутри каталога у вас будет находиться индексный файл, который отображается по умолчанию, когда пользователь переходит в каталог в своем браузере. Если пользователь вводит

www.ellumitechacademy.com/courses

веб-сервер будет искать в каталоге файл index.html:

www.ellumitechacademy.com/courses/index.html

Если файл index.html отсутствует, веб-сервер попытается отобразить список файлов, либо вы увидите сообщение об ошибке (Рис. 1-5).

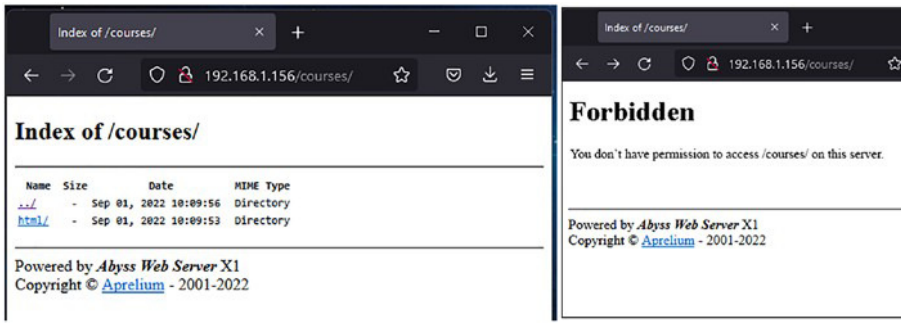


Рис. 1-5. *Листинг каталогов веб-сервера*

HTML5

HTML5 обеспечивает независимость отображения от устройств, а это означает, что веб-сайты можно разрабатывать для всех типов платформ, от ПК до смартфонов, без необходимости бесконечно устанавливать плагины в браузер или разрабатывать несколько версий веб-сайта для мобильных устройств, как мы видим это на Рис. 1-6.

Рис. 1-6. *Веб-сайт на разных устройствах*

В HTML5 также появились некоторые новые теги для управления структурой страницы, такие как <section>, <head>, <nav>, <aside> и <footer>, а также некоторые теги для обработки мультимедиа, такие как <audio> или <video>.

Некоторые новые функции HTML5 мы рассмотрим позже в этом руководстве.

Что такое CSS?

Каскадные таблицы стилей (CSS) используются для определения и

настройки стилей и макета ваших веб-страниц. Это означает, что вы можете создавать таблицы стилей для изменения дизайна, макета и реакции на различные размеры экрана на различных устройствах, от компьютеров до смартфонов.

В CSS селекторы определяют, к какой части HTML-разметки применяется стиль. Селектором может быть стиль заголовка H1, тег содержимого или тег абзаца.

Итак, вы увидите селектор, скажем, H1, а внутри фигурных скобок вы увидите блок объявлений, в котором вы объявляете свои стили для этого селектора (Рис. 1-7).

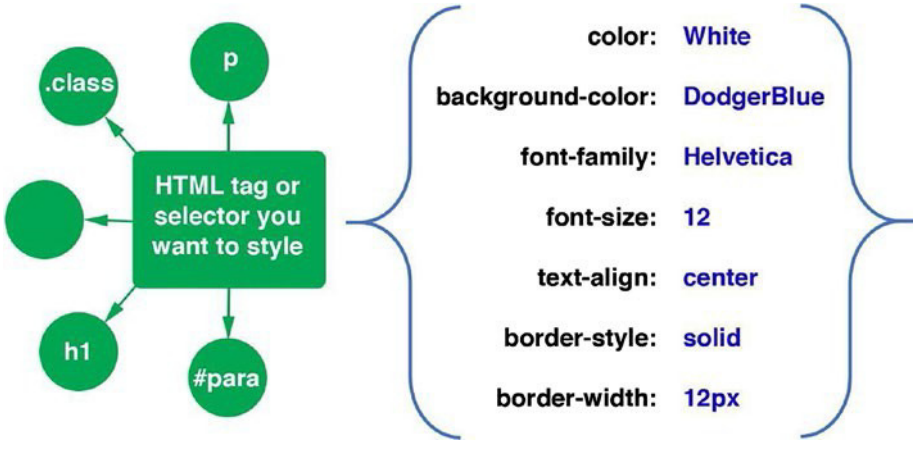


Рис. 1-7. *Анатомия CSS-селектора*

Вы можете добавить объявления CSS в раздел <head> вашего HTML-документа между тегами <style>…</style> или добавить объявления CSS в отдельный файл style.css и добавить ссылку в раздел <head>. вашего HTML-документа, используя

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

Это лучший способ, поскольку он позволяет вам менять стили в одном месте, а не на каждой создаваемой вами HTML-странице.

Мы более подробно рассмотрим CSS позже, в Главе 4 этой книги.

Хостинг

Для хостинга вашего сайта у вас есть три варианта:

- Выделенный хостинг, при котором сервер расположен в вашей школе или предоставляется в ваше пользование. Его можно использовать для разработки, а в некоторых случаях и для активного веб-сайта в зависимости от услуги.
- Настройка персонализированного веб-сервера на вашем компьютере. Это подходит только для тестирования и разработки и не предназначено для размещения действующего веб-сайта.
- Управляемый хостинг или веб-хостинг, который представляет собой услугу, управляемую провайдером веб-хостинга: это то, что вы будете использовать для размещения общедоступного действующего веб-сайта.

Установка нашего веб-сервера

В данном руководстве мы будем использовать персональный веб-сервер. Это поможет вам настроить среду разработки, которую вы сможете использовать на своем компьютере для тестирования вашего веб-сайта, не имея к нему доступа в Интернете.

Abyss превращает ваш компьютер в полнофункциональный веб-сервер. Чтобы загрузить веб-сервер Abyss, откройте веб-браузер и перейдите по адрсу

aprelium.com/downloads

С левой стороны нажмите “Free Download” (Рис. 1-8).

Рис. 1-8. *Веб-страница загрузки*

Выберите версию Windows (или версию Mac, если вы используете Mac) (Рис. 1-9).

Рис. 1-9. *Страница загрзки веб-сервера Abyss*

Откройте каталог загрузок, затем дважды щелкните только что загруженный пакет программного обеспечения. Нажмите “Agree” на странице лицензионного соглашения.

Отмените выбор компонентов, которые вы не хотите устанавливать. Выберите все компоненты, кроме “Abyss Web Server (32-bit).” “Start Menu Shortcuts” позволяет добавлять ярлыки веб-сервера Abyss в меню «Пуск», а “Documentation” устанавливает файлы справки (Рис. 1-10).

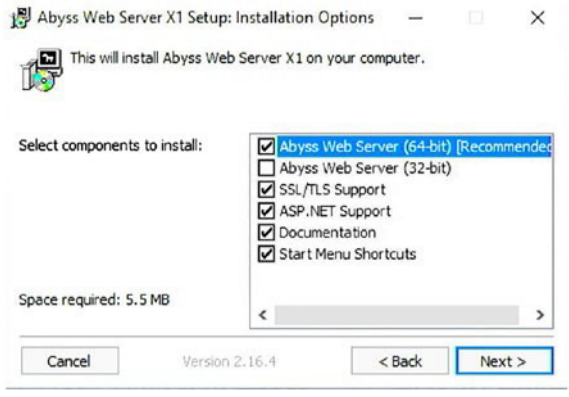


Рис. 1-10. *Установка веб-сервера*

Выберите каталог, в который вы хотите установить файлы веб-сервера Abyss. Нажмите “Install” (Рис. 1-11).

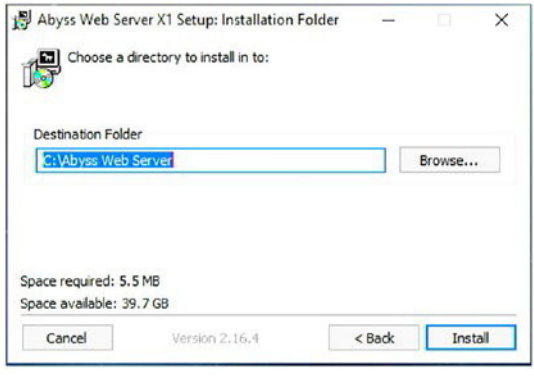


Рис. 1-11. *Выбор каталога для установки веб-сервера*

В меню «Пуск» нажмите “Abyss Web Server X1” (Рис. 1-12).

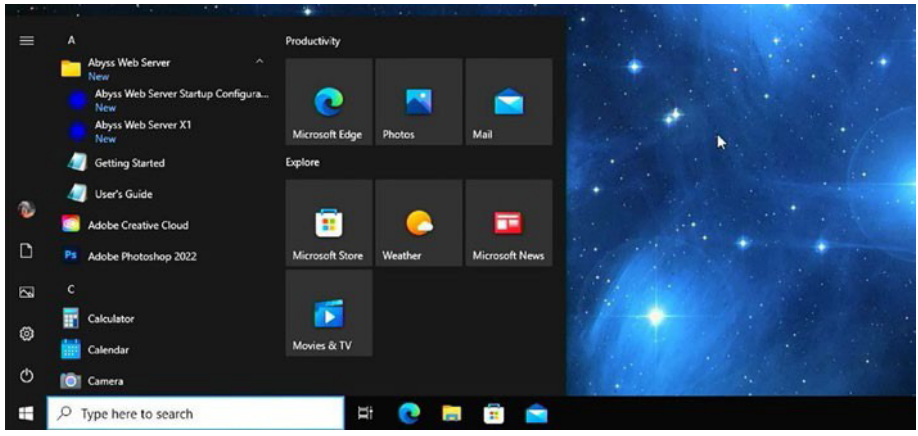


Рис. 1-12. *Меню Пуск*

В вашем веб-браузере откроется мастер настройки (Рис. 1-13). Выберите ваш язык.

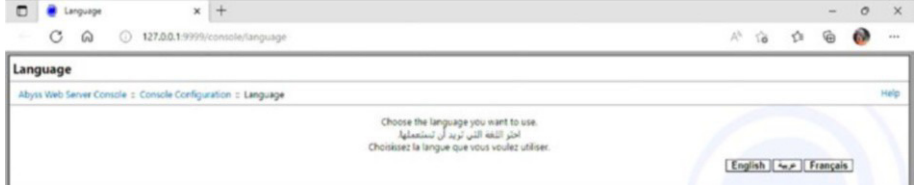


Рис. 1-13. *Мастер настройки*

Создайте свои данные для входа – не забудьте их. Введите имя пользователя для входа (например, admin) и пароль (Рис. 1-14). Нажмите OK, когда закончите.

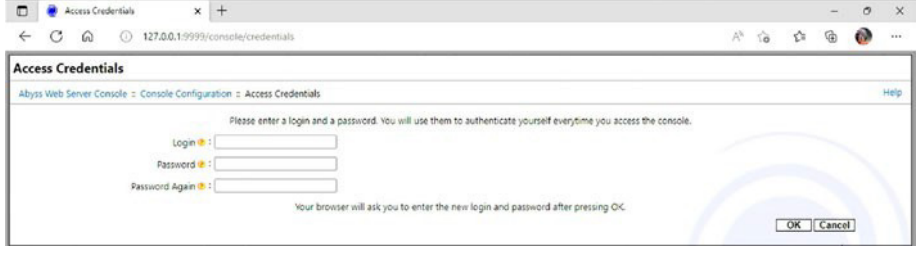


Рис. 1-14. *Учетная запись*

Когда браузер запросит у вас имя пользователя и пароль, введите имя пользователя и пароль, которые вы выбрали ранее (Рис. 1-15).

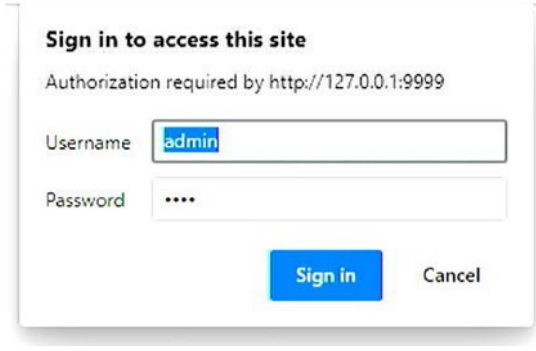


Рис. 1-15. *Ввод учетных данных*

Запуск веб-сервера

Мы настроили сервер так, чтобы он не запускался автоматически при входе в Windows, поскольку это может представлять угрозу безопасности. Лучше всего запускать веб-сервер только тогда, когда он вам нужен для тестирования вашего веб-сайта.

После установки веб-сервера вам нужно будет запустить веб-сервер, прежде чем вы сможете что-либо сделать.

Чтобы запустить сервер, откройте меню «Пуск», затем нажмите “Abyss Web Server X1” (Рис. 1-16).

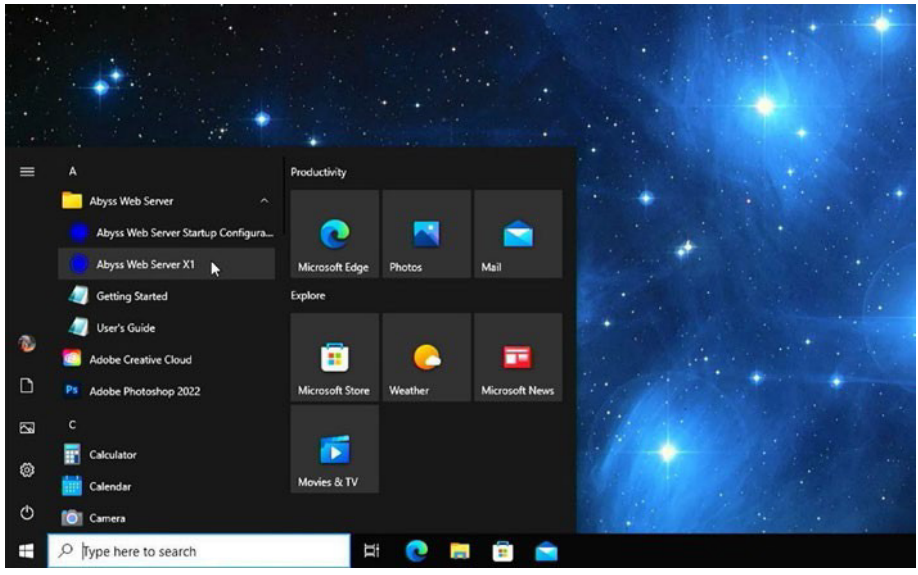


Рис. 1-16. *Зауск веб-сервера Abyss*

Сервер появится в системной области в правом нижнем углу (Рис. 1-17).

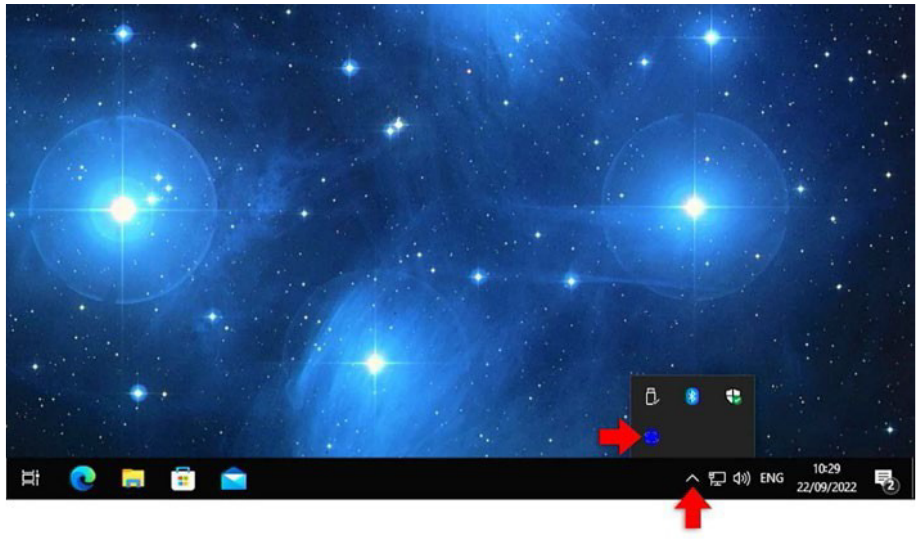


Рис. 1-17. *Веб-сервер Abyss в системной области*

Сохранение ваших веб-страниц

Вы можете сохранить свои веб-страницы на веб-сервере Abyss на своем локальном компьютере или использовать FTP для загрузки их на веб-хостинг, если у вас есть такая возможность. В этом курсе мы сохраним наши страницы на веб-сервере Abyss на локальном компьютере.

Локальная машина

Если вы установили веб-сервер Abyss на свой локальный компьютер, любые страницы, которые вы разрабатываете на своем веб-сайте, будут сохранены в следующей папке:

c:\abyss web server\htdocs

Вы найдете папку в проводнике на диске C (Рис. 1-18).

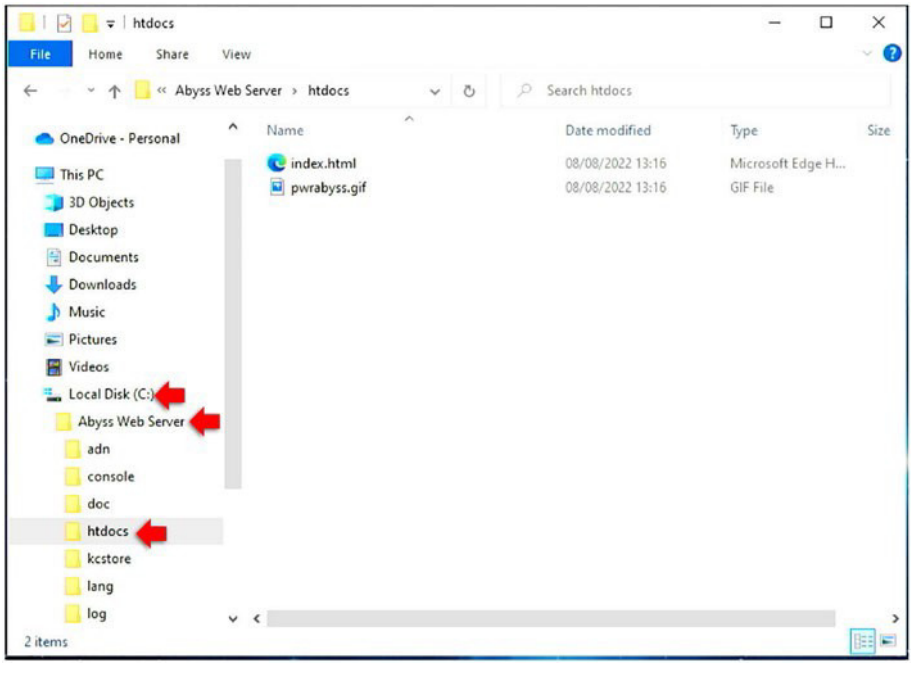


Рис. 1-18. *Страницы веб-сервера Abyss*

После запуска сервера вы сможете получить доступ к своей веб-странице из веб-браузера, перейдя по адресу

http://localhost/**pagename.html**

либо

http://127.0.0.1/**pagename.html**

pagename.html — наименование HTML-документа, который вы хотите просмотреть.
Это может быть

http://localhost/**index.html**
http://localhost/**store.html**

Использование веб-хостинга

Если вы используете веб-хостинг где-то еще, вам нужно будет использовать FTP-клиент для подключения к серверу и загрузки файлов. Вам необходимо будет получить имя пользователя и пароль, а также адрес веб-сервера у вашего хостинг-провайдера.

Если вы используете Windows или Mac, хорошим FTP-клиентом является FileZilla. Вы можете скачать это на следующем веб-сайте:

filezilla-project.org/download.php

После загрузки и установки на свой компьютер запустите программу. В меню "File" выберите "Site Manager" (Рис. 1-19).

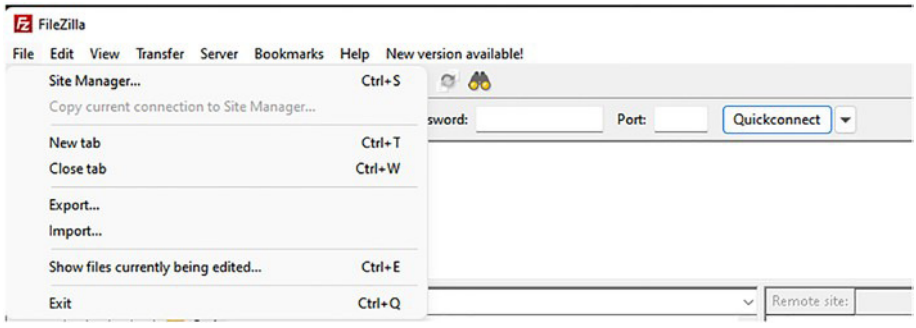


Рис. 1-19. *Менеджер сайта с использованием FileZilla*

Нажмите “New site,” затем введите имя хоста FTP или IP-адрес, затем добавьте свое имя пользователя и пароль (Рис. 1-20).

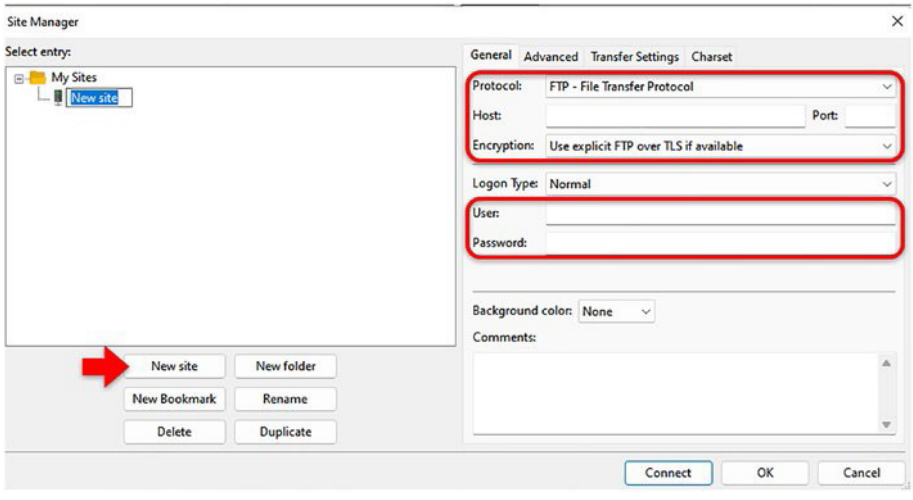


Рис. 1-20. *FTP и процесс входа в систему*

Нажмите Click, чтобы начать.

Как только соединение с сервером будет установлено, вы попадете на главный экран.

На левой панели перейдите к папке, в которой вы сохраняете все свои HTML-файлы. На панели справа перейдите к папке htdocs или public_html на веб-сервере (Рис. 1-21).

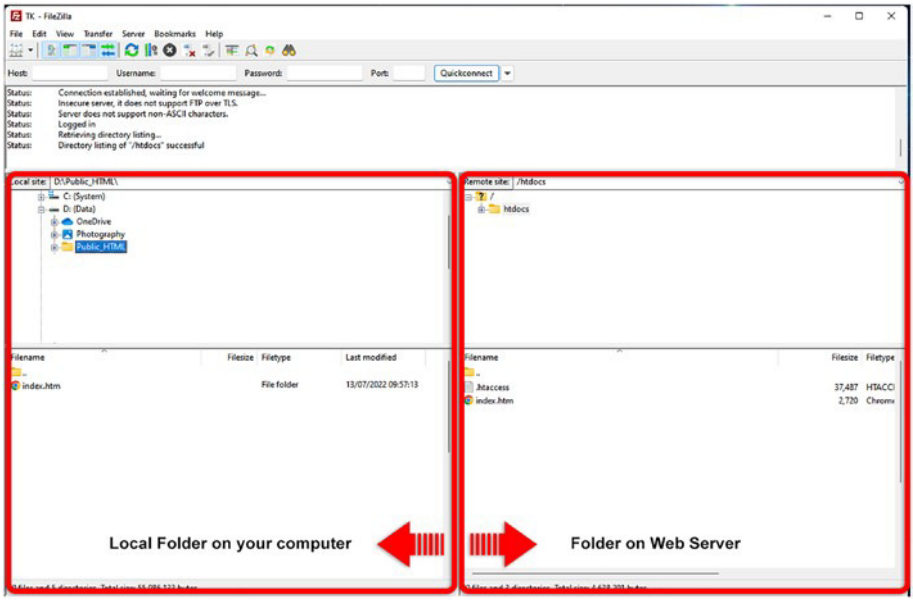


Рис. 1-21. *Файлы на локальном компьютере и веб-сервере*

Чтобы загрузить файлы, выберите их на левой панели, затем щелкните на них правой кнопкой мыши. Во всплывающем меню выберите “Upload” (Рис. 1-22). Чтобы загрузить файлы, выберите их на правой панели, затем щелкните выделение правой кнопкой мыши. Во всплывающем меню выберите “Download”.

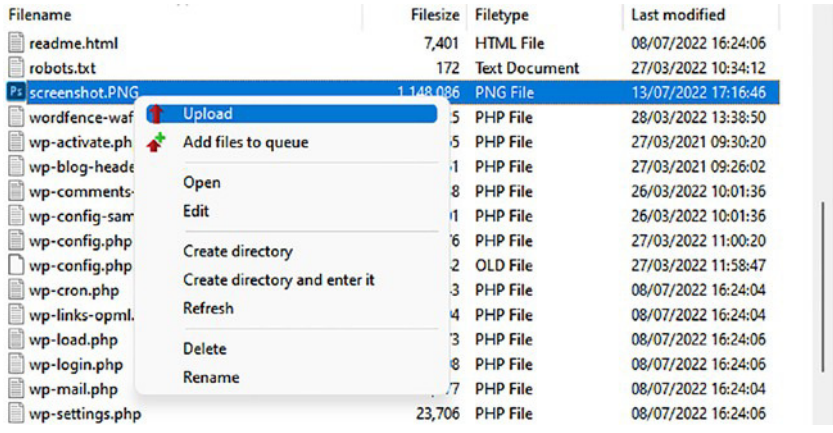


Рис. 1-22. *Загрузка и скачивание файлов*

Для получения более подробной информации о том, как использовать FileZilla, перейдите сюда:

wiki.filezilla-project.org/Documentation

Инструменты разработки и редакторы кода

Выбор подходящего инструмента зависит от личных предпочтений и типа приложения, которое вы собираетесь разрабатывать. Доступно много разных инструментов. Вы можете использовать IDE, которая представляет собой интегрированную среду разработки, например Adobe Dreamweaver, Brackets или VS Code.

IDE — это программное приложение, состоящее из редактора исходного кода с подсветкой синтаксиса, облегчающей чтение кода, а также встроенных инструментов, помогающих разрабатывать код, и отладчика, помогающего находить ошибки. Все они интегрированы в одно приложение, отсюда и название «интегрированная среда разработки».

На Рис. 1-23 мы видим Dreamweaver. Вы можете видеть, что у него есть предварительный просмотр вверху, код внизу, а также

различные другие инструменты и параметры, которые помогут вам написать свой код.

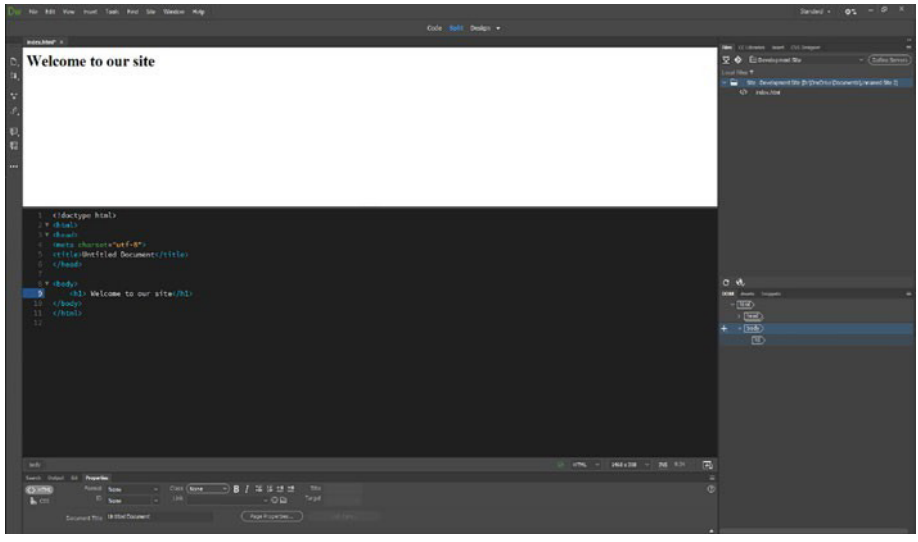


Рис. 1-23. *IDE Adobe Dreamweaver*

Другая популярная среда разработки — Visual Studio Code или сокращенно VS Code. Вы можете скачать VS Code со следующего сайта:

code.visualstudio.com

На Рис. 1-24 мы видим слева VS Code, отображающийй наш HTML-код, а справа открыт предварительный просмотр браузера, показывающий результат HTML-кода на выходе.

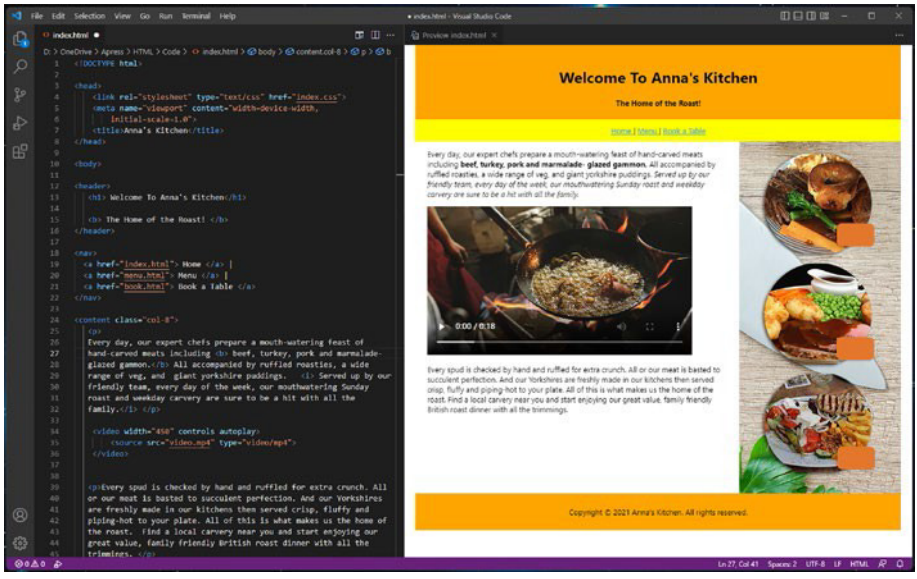


Рис. 1-24. *IDE VS Code*

Еще один редактор кода, который стоит попробовать, — Brackets. Brackets — бесплатный редактор, который можно скачать с сайта разработчика:

www.brackets.io

На Рис. 1-25 вы можете видеть слева открытое окно Brackets, которое представляет собой довольно приятный небольшой редактор для написания кода. С правой стороны вы можете в реальном времени открыть предварительный просмотр, чтобы увидеть, как выглядит ваша страница во время создания кода.

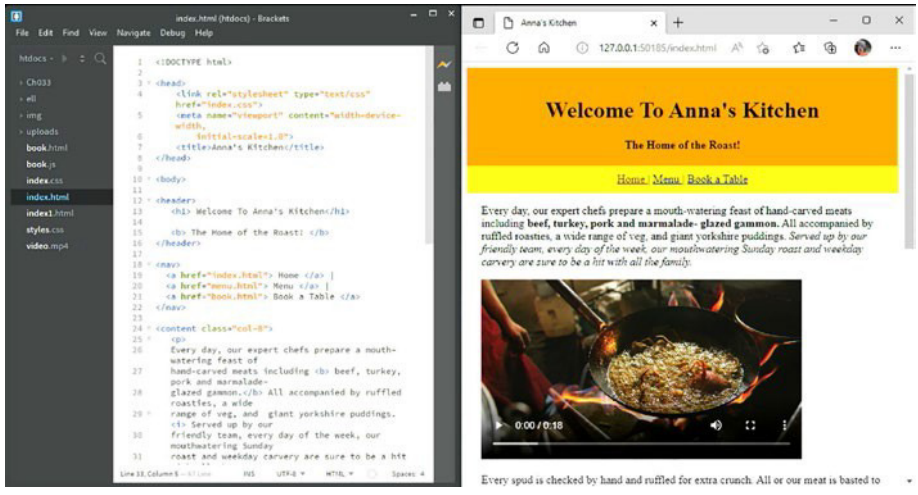


Рис. 1-25. *Приложение Brackets с предварительным просмотром в браузере*

Вы можете использовать любой из этих инструментов для написания кода. Некоторые из этих IDE могут быть довольно сложными, поэтому, пока вы учитесь, я предлагаю вам использовать Блокнот и писать код вручную, чтобы вы могли понять структуру и значение, не отвлекаясь.

На протяжении всей книги мы будем использовать Блокнот/TextEdit для написания кода, как показано на Рис. 1-26; однако, если хотите, вы можете использовать любой редактор кода, который вам нравится, например VS Code.

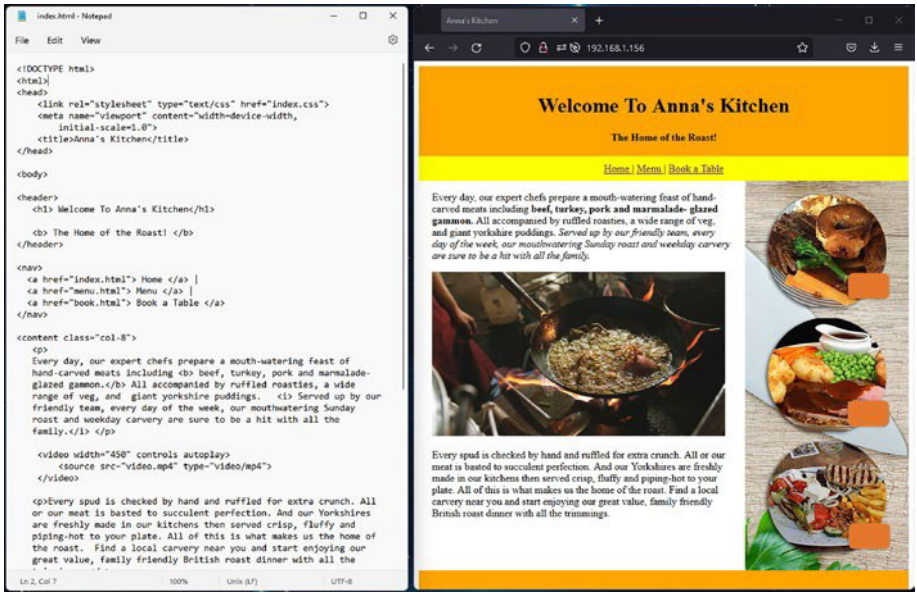


Рис. 1-26. *Редактор кода Блокнота и предварительный просмотр в браузере*

Практичский пример

В этом примере мы собираемся изучить, как работают веб-серверы.

На Рис. 1-27 на машине справа установлен наш веб-сервер. Веб-сервер — это программа, которая запускается на компьютере и обслуживает веб-страницы (например, Abyss, который мы установили ранее). Программное обеспечение веб-сервера привязано к порту. Порт — это номер, используемый для идентификации службы, работающей на машине. В данном случае служба представляет собой веб-сервер и привязана к порту 80. Этот компьютер подключен к небольшой сети с помощью кабелей Cat5 через сетевой коммутатор.

Рис. 1-27. *Машина с установленным веб-сервером*

Веб-сервер, работающий на машине справа, указывает на каталог public_html или htdocs, хранящийся на жестком диске машины. Здесь мы видим, что у нас на нашем сервере есть файл index.html в каталоге public_html (Рис. 1-28). Он называется корневым документом.

Рис. 1-28. *Корень документов*

Каждый компьютер в сети имеет уникальный IP-адрес, который идентифицирует устройство в сети. Веб-сервер имеет IP-адрес 192.168.0.100 и привязан к порту 80 (по умолчанию для незашифрованного веб-трафика). Вы можете увидеть следующую сводную информацию о конфигурации:

- IP-адрес: 192.168.0.100
- Порт: 80

- Корень документов: /var/www/public_html

Давайте добавим еще один компьютер. Компьютер слева называется клиентом и представляет собой ноутбук под управлением Windows с установленным веб-браузером (Рис. 1-29). Браузер может быть Edge, Firefox или Chrome.

Рис. 1-29. *Клиент с доступом к веб-серверу*

Этот ноутбук подключается к веб-серверу, используя IP-адрес, выделенный машине, на которой работает веб-сервер. На ноутбуке мы можем ввести этот IP-адрес в адресную строку в верхней части браузера: 192.168.0.100 (Рис. 1-30).

Рис. 1-30. *IP-адрес*

Ноутбук подключится к веб-серверу, используя IP-адрес сервера через порт 80 (Рис. 1-31).

Рис. 1-31. *Подключение к веб-серверу*

На ноутбуке соединению присваивается номер порта от 49 152 до 65 535, чтобы возвращаемый трафик с веб-сервера можно было идентифицировать как принадлежащий тому же соединению (Рис. 1-32).

Рис. 1-32. *Номер порта, назначенный соединению*

IP-адрес и номер порта образуют сокет. Будет существовать один сокет на сервере и один на ноутбуке (клиенте). Каждый сокет уникален и двунаправлен, поэтому приложения могут отправлять и получать данные (Рис. 1-33).

Рис. 1-33. *Сокет*

Затем браузер на ноутбуке (клиенте) считывает HTML-код и создает веб-страницу, которую вы и видите на экране.

Возможно, вы заметили, что при посещении веб-сайта вы не вводите строку из цифр, а вводите доменное имя. Проблема в том, что компьютеры не понимают доменные имена, а понимают только IP-адреса, поэтому нам нужно преобразовать имена в адреса.

Для этого нам нужно добавить в сеть еще один сервер, называемый DNS-сервером. Этот сервер преобразует привычные для нас доменные имена в IP-адреса.

Когда вы вводите имя домена в свой браузер, например, elluminetpress.com, ваш компьютер (ноутбук) отправляет имя домена DNS-серверу. DNS-сервер отвечает IP-адресом (например, 192.168.0.100) (Рис. 1-34).

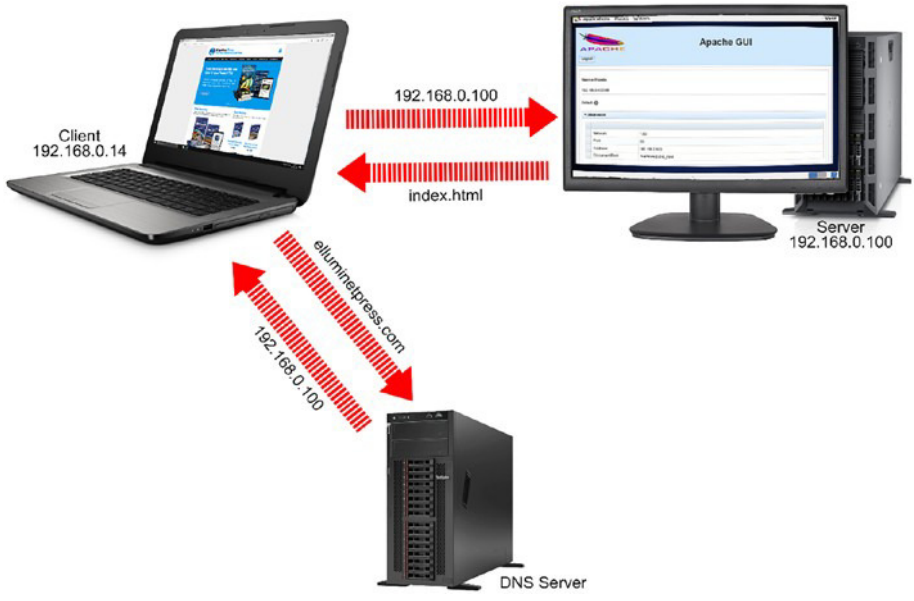


Рис 1-34. *Подключение компьютера к веб-серверу*

Затем ваш компьютер (то есть ноутбук), как и раньше, использует этот IP-адрес для подключения к веб-серверу.

Упражнение

1. Настройте свой личный веб-сервер на своем компьютере, либо получите доступ к веб-хостингу для размещения ваших HTML-файлов.
2. Выберите текстовый редактор для редактирования кода, например Блокнот.
3. Что такое гипертекст?
4. Что такое URL-адрес?
5. Что такое HTML?

6. Что такое CSS?
7. Что такое веб-сервер?
8. Каково назначение файла index.html?
9. Что такое IP-адрес?
10. Каково назначение DNS-сервера?

Заключение

- Все веб-страницы связаны друг с другом с помощью интерактивного текста или изображений, называемых гиперссылками.
- Редакторы кода и IDE
  - VS Code
  - Dreamweaver
  - Brackets
  - Текстовый редактор (Блокнот, TextEdit и т.д.)
- Веб-сайт и его страницы хранятся или размещаются на веб-сервере.
- DNS-сервер преобразует доменные имена в IP-адреса.
- IP-адрес – это уникальный адрес, который идентифицирует устройство в сети Интернет.

ГЛАВА 2

Введение в HTML

В этой главе мы рассмотрим основы HTML-документов.
Базовая структура HTML-документа состоит из трех частей:

- Объявление типа документа вверху
- Заголовок документа
- Содержимое докумена

Структура HTML-страницы

На Рис. 2-1 мы можем видеть, что HTML-страница — это текстовый файл, содержащий элементы и информацию, которую веб-браузер использует для отображения веб-страницы. Статические веб-страницы имеют расширение файла **.htm** или чаще **.html**.

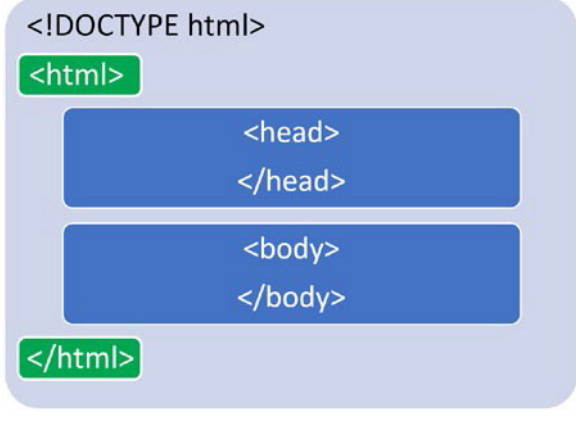


Рис. 2-1. *Структура HTML-документа*

Вверху в первой строке у нас присутствует объявление типа документа.

Ниже у нас находится первый элемент <html>. Он определяет начало HTML-страницы.

Внутри элементов <html> у нас следует элемент <head>. Он содержит информацию о странице, а также заголовок документа.

Еще ниже у нас находится элемент <body>. Здесь определяется основная часть документа. Это то, что отображается в окне браузера.

Наконец, нам нужно закрыть элемент <html>. Он означает конец документа.

Давайте немного подробнее рассмотрим пример. Здесь вы можете видеть упрощенную веб-страницу, разбитую на самые основные элементы (Рис. 2-2).

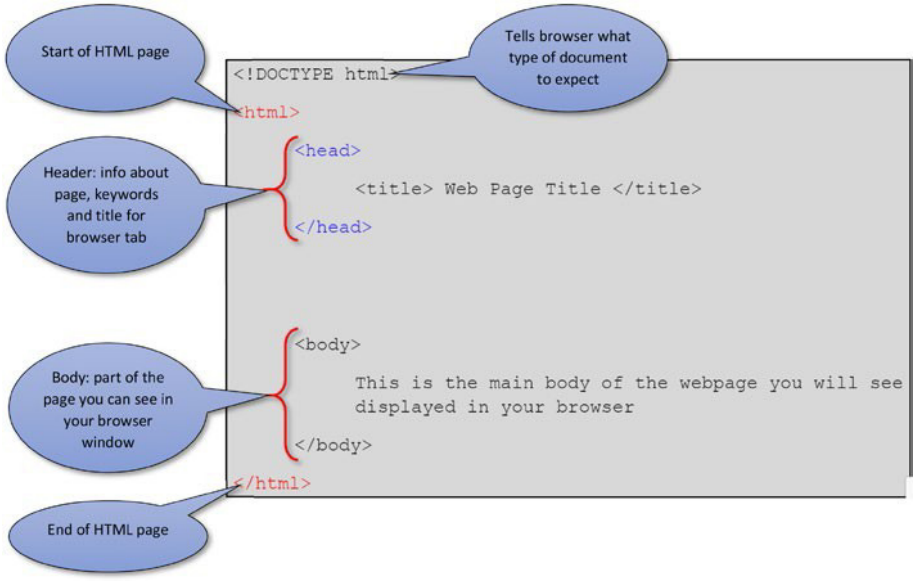


Рис. 2-2. *Пустой HTML-документ*

<!DOCTYPE html> указывает, какой тип документа может ожидать веб-браузер, в данном случае HTML5.

Элемент **<html>** содержит весь HTML-код и определяет начало HTML-страницы. Вы также можете указать язык страницы, добавив атрибут lang:

<html **lang="en"**> , чтобы указать английский язык

<html **lang="es"**> , чтобы указать испанский язык

<html **lang="fr"**> , чтобы указать французский язык

<html **lang="de"**> , чтобы указать немецкий язык

В **<head>** документа вы встретите

<title>: здесь мы размещаем наименование страницы, которое будет отображаться в верхней части окна или вкладки браузера.

<style>: здесь мы определяем внутреннюю информацию о стиле для HTML-документа с помощью CSS. См. Главу 4.

<link>: ссылка на внешнюю таблицу стилей. См. Главу 4.

<script>: используется для определения клиентских сценариев, таких как JavaScript. См. Главу 8.

<meta>: здесь хранится информация о документе – кодировка символов, наименование, описание и т.д. См. раздел «Метаданные» далее в этой главе.

Элемент **<body>** содержит все элементы, и в нем записывается основной контент для отображения на веб-странице.

Вы также можете встретить следующие элементы:

<!-- … -->

Такие элементы содержат комментарий для разработчика и игнорируются браузером. Комментарии полезны для

документирования вашего кода и объяснения его функций.

Структура элемента HTML

Технически элемент HTML состоит из начального тега, атрибутов элемента, видимого бита или содержимого и конечного тега. Тег HTML используется для обозначения начала или конца элемента.

Давайте подробнее рассмотрим, как создаются HTML-элементы. Элементы начинаются с открывающего тега и заканчиваются закрывающим тегом. Сами теги начинаются и заканчиваются угловыми скобками <>.

В примере на Рис. 2-3 давайте рассмотрим элемент привязки. Этот элемент создает гиперссылку на другую веб-страницу. HTML-элемент начинается с открывающего тега и заканчивается закрывающим тегом.

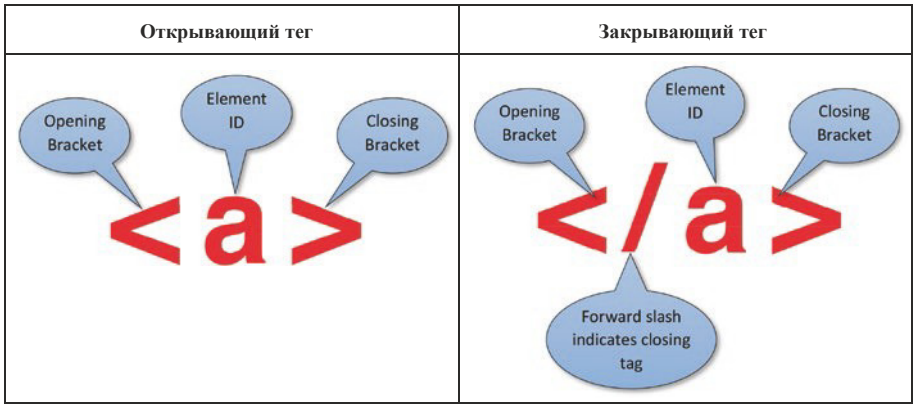


Рис. 2-3. *HTML-теги открытия и закрытия*

Бит, видимый пользователю, находится между двумя тегами.

Открывающий тег HTML часто содержит некоторые атрибуты, которые определяют свойства элемента HTML и используются для управления форматированием, размером, ссылками на страницы и т. д. Он помещается внутри открывающего тега элемента. Например, см. Рис. 2-4.

Рис. 2-4. *Структура HTML-тегов*

Атрибуты HTML состоят из двух частей: наименования и значения.

- **name** (наименование) — это атрибут, который вы хотите установить. Например, элемент привязки <a> содержит атрибут с наименованием «href», который указывает адрес страницы, на которую вы хотите создать ссылку.
- **value** (значение) — это то, что вы хотите установить для атрибута, и значние всегда заключено в кавычки. В этом примере страница, на которую мы хотим создать ссылку, называется «about.html».

Давайте посмотрим на пример. Здесь мы хотим добавить на сайт ссылку “About Us”. Элемент привязки обозначается буквой «a» и записывается как

<a href = "about.html"> About Us ****

Давайте разберем элемент и посмотрим, как он работает.

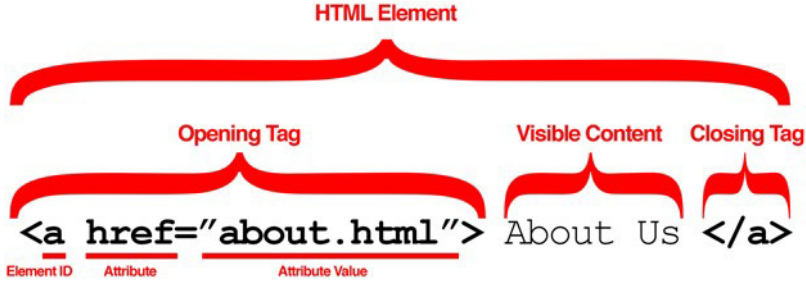


Рис. 2-5. *HTML-элемент*

Элемент начинается с открывающей угловой скобки <, за

которой следует идентификатор элемента, который мы хотим использовать, в данном случае «a» для привязки. После этого мы добавляем любые атрибуты, как это показано на Рис. 2-5.

Атрибуты содержат дополнительную информацию. Атрибуты имеют форму открывающего тега, внутри которого размещается дополнительная информация. Например, в теге HTML, показанном выше, «href» — это атрибут, а «about.html» — это значение атрибута.

Закрываем открывающий тег угловой скобкой >.

После открывающего тега мы добавляем видимый бит, который пользователь увидит на веб-странице “About Us”.

Закрывающий тег содержит косую черту перед идентификатором элемента, в данном случае:

На Рис. 2-6 слева мы видим элемент привязки в HTML-документе. Текст “About Us” отображается на веб-странице в окне браузера и он связан со страницей about.html.

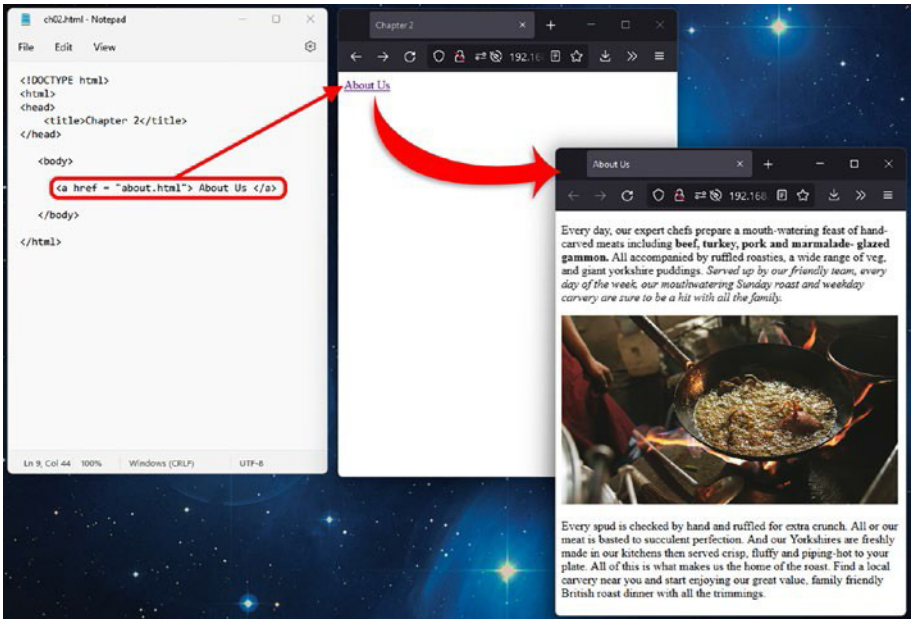


Рис. 2-6. *Элемент привязки*

Аналогично для элемента изображения: мы начинаем с открывающего тега <img>, затем добавляем атрибут src, содержащий изображение, которое мы хотим отобразить:

```
<img  src="profilepic.jpg">
```

Источник изображения (src) является атрибутом открывающего тега <img>. Обратите внимание, что элемент <img> не имеет закрывающего тега. Это называется пустым или самозакрывающимся элементом.

Метаданные

Метаданные — это дополнительная информация о HTML-документе. Мета-элементы можно использовать для описания свойств HTML-документа, таких как автор, дата и описания содержимого. Метаданные используются браузерами для определения способа отображения контента, а поисковые системы, такие как Bing или Google, — для определения того, о чем веб-страница.

<meta> всегда находится внутри <head> HTML-страницы, как вы можете это видеть ниже:

```
<head>
  <meta charset="UTF-8">
  <meta name="description" content="Enjoy succulent meats…">
  <meta name="author" content="Anna Wilson">
</head>
```

Мета-элемент имеет различные атрибуты, такие как кодировка и наименование.

Мета-кодировка определяет кодировку символов для HTML-документа, обычно это UTF-8.

Мета-наименование определяет наименования метаданных, такие как описание контента, которое отображается в результатах поиска страницы, ключевые слова, идентифицирующие контент, и автор страницы.

Упражнение

1. Откройте новый текстовый файл и сохраните его как ch02.html.
2. Напишите базовую структуру HTML-документа.
3. Что такое HTML-тег?
4. Что такое HTML-элемент?
5. В чем разница между HTML-тегом и HTML-элементом?
6. Что такое метаданные?
7. Для чего нужен раздел <head> в HTML-документе? Какие еще элементы вы можете включить в HTML-документ?

Заключение

- Элемент HTML состоит из начального тега, атрибутов элемента, видимого бита или содержимого и конечного тега. Тег HTML используется для обозначения начала или конца элемента.
- <!DOCTYPE html> указывает, какой тип документа может ожидать веб-браузер, в данном случае HTML5.
- Элементы заголовка содержат информацию о странице, а также заголовок документа. Они также содержат другие элементы, такие как title для указания заголовка страницы для окна браузера, style для включения стилей CSS, script для включения любых сценариев JavaScript и meta для включения метаданных.
- Элементы body содержат основную часть документа. Это то, что вы видите в окне браузера.

ГЛАВА 3

Итак, начнем с HTML

В этой главе мы собираемся создать очень простую домашнюю страницу для веб-сайта нашего ресторана, используя общие элементы форматирования текста, и добавив несколько изображений, ссылок, таблиц и списков.

Это создаст для вас основу и базовую структуру веб-страницы с использованием HTML, которую вы сможете построить позже.

Мы пройдем процесс, начиная с чистого HTML-документа, а затем создадим домашнюю страницу, используя элементы HTML.

Мы также рассмотрим, как использовать атрибуты стиля каждого элемента HTML. Вы можете применять стили непосредственно в HTML-коде каждый раз, когда хотите использовать стиль, например шрифты, цвет текста и цвет страницы. Однако это несколько громоздкий способ, и становится очень сложно придерживаться ему в долгосрочной перспективе, особенно в крупных проектах с большим объемом кода. Гораздо лучший способ — использовать таблицу стилей или CSS для установки шрифтов, цвета текста и цвета страницы. Это позволит вам определить все ваши стили один раз и в одном месте, обычно в файле styles.css. Затем вы можете вызывать файл CSS из своего HTML-кода, который мы рассмотрим в следующей главе.

А пока давайте сосредоточимся на HTML-коде.

Конфигурирование

Для упражнений в этой главе мы будем использовать Блокнот и веб-браузер. Мы будем сохранять наши HTML-файлы в папке htdocs на нашем личном веб-сервере, который мы установили в Главе 1.

Откройте текстовый редактор. Я буду использовать Блокнот. Здесь мы введем наш код. Мы начнем с структуры HTML-документа. Например, см. Рис. 3-1.

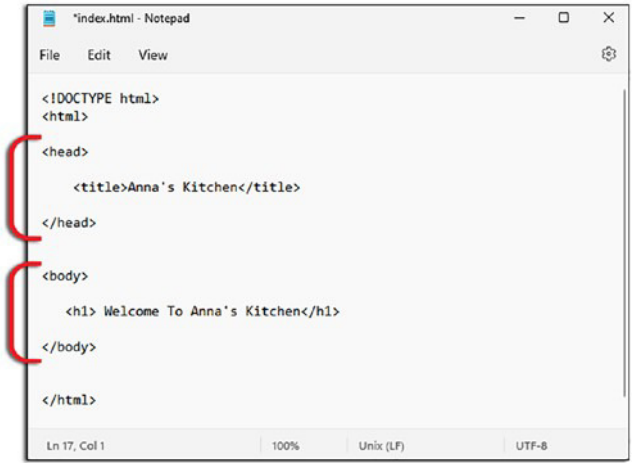


Рис. 3-1. *Структура HTML-документа*

Чтобы сохранить файл, перейдите в меню “File” (Файл), затем нажмите “Save as” («Сохранить как») (Рис. 3-2).

Рис. 3-2. *Save As в Блокноте*

В диалоговом окне “Save As” («Сохранить как») перейдите в папку «Abyss Web Server» на диске C, затем выберите «htdocs» (Рис. 3-3).

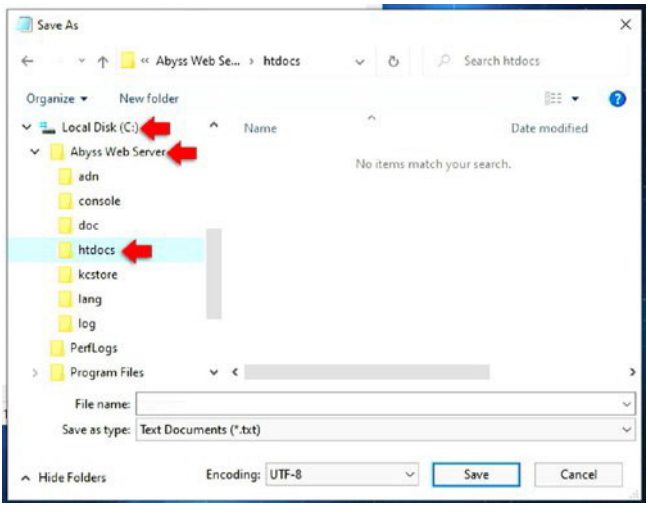


Рис. 3-3. *Сохранение в Web Server*

В поле “File name” («Имя файла») введите наименование вашей веб-страницы:

index.html

Убедитесь, что установлено расширение файла **.html** , а для параметра “**Save as type**” («Тип файла») выбрано “All files” («Все файлы») (Рис. 3-4).

Рис. 3-4. *Сохранение в Abyss Web Server*

При работающем персональном веб-сервере Abyss откройте веб-браузер. Введите следующее в адресную строку браузера:

http://127.0.0.1/index.html

index.html — это файл, который мы хотим просмотреть. Чтобы просмотреть любые другие созданные вами HTML-файлы, просто замените это имя файла на имя нужного файла.

Расположите окна рядом; поместите Блокнот слева, а веб-браузер рядом с ним справа. Возможно, вам придется изменить размер окон.

Каждый раз, когда вы вносите какие-либо изменения в файл в Блокноте, вам необходимо сохранить его, а затем щелкнуть значок обновления в веб-браузере..

Давайте начнем.

Элементы форматирования текста

Как вы можете видеть на иллюстрации на Рис. 3-5, текст в тегах <title> отображается на вкладке заголовка в веб-браузере, а все,

что находится между тегами <body>, отображается в окне браузера.

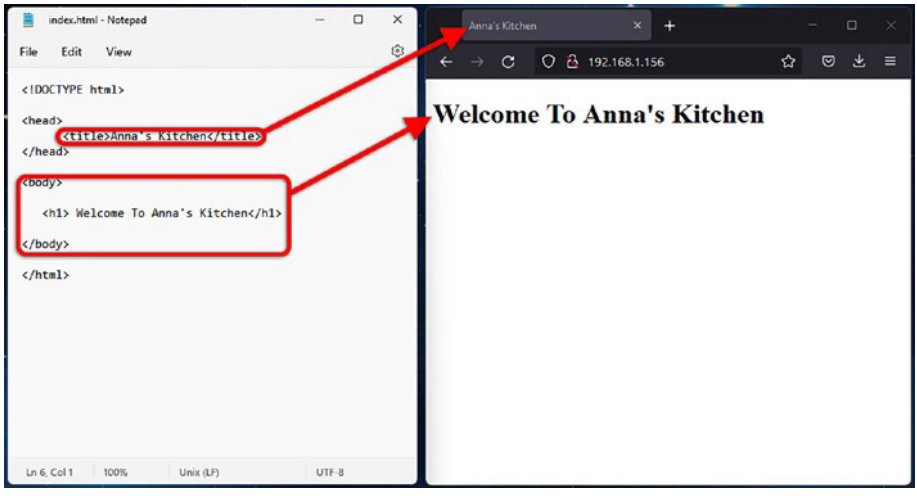


Рис. 3-5. *Заголовок и основной текст*

Элементы HTML обозначают части вашей веб-страницы, такие как заголовки, форматирование текста жирным или курсивом, абзацы, изображения, ссылки и таблицы.

Давайте рассмотрим некоторые простые элементы форматирования.

Стиль основного заголовка

<h1>...</h1>

Стиль подзаголовка

<h2>...</h2>

Жирный текст

...

Курсивный текст

...

Текст абзаца

...

Элементы HTML начинаются и заканчиваются тегом HTML и обычно встречаются парами, и вам нужно будет выделить фрагмент текста или слова, используя начальный и конечный теги.

Например:

<h1> This is the main page heading **</h1>**

Заголовки

Давайте добавим некоторые из этих элементов на нашу веб-страницу. Начните с заголовка, используя тег <h1>.

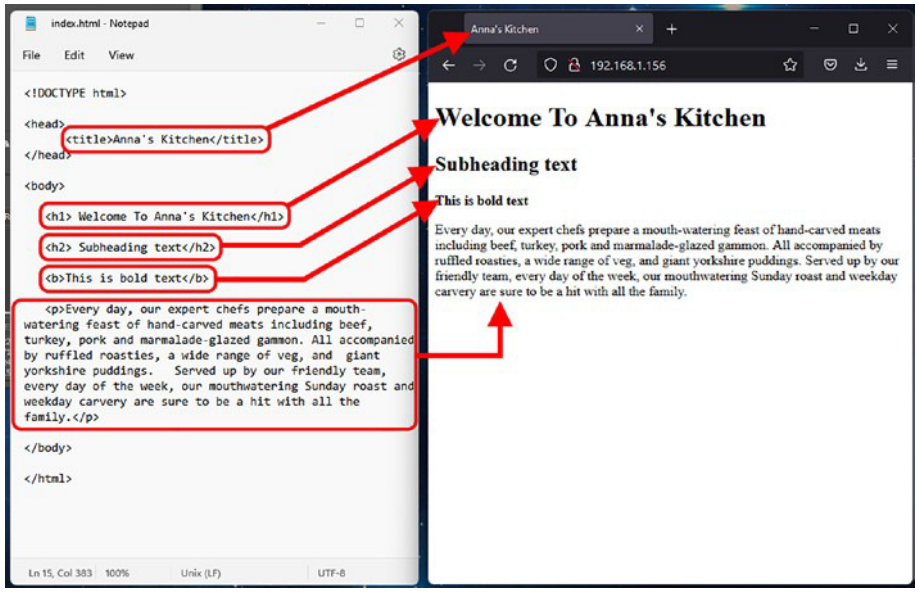


Рис. 3-6. *Текстовый редактор HTML*

На Рис. 3-6 HTML-документ открыт в текстовом редакторе слева. Тот же документ открыт в веб-браузере справа, и вы можете увидеть эффект, который каждый элемент оказывает на текст, что отмечено красными стрелками.

Браузеры не отображают элементы HTML, а используют их для форматирования содержимого страницы в соответствии с их функцией.

Параграфы

Вы также можете добавлять абзацы. Лучше всего добавлять все абзацы между тегами ... :

** ** Каждый день наши опытные повара готовят аппетитный мясной стол, включая говядину, индейку, свинину и окорок. И все это в сопровождении жаркого из большого разнообразия овощей и гигантских йоркширских пудингов. Наше аппетитное воскресное жаркое и мясные блюда в будние дни подаются нашей дружной командой каждый день недели и обязательно понравятся всей семье.** **

Жирный текст

Вы также можете сделать текст жирным или насыщенным. Просто окружите слово или слова тегами:

**** говядина, индейка, свинина и окорок****
**** говядина, индейка, свинина и окорок****

Посмотрите на пример (Рис. 3-7).

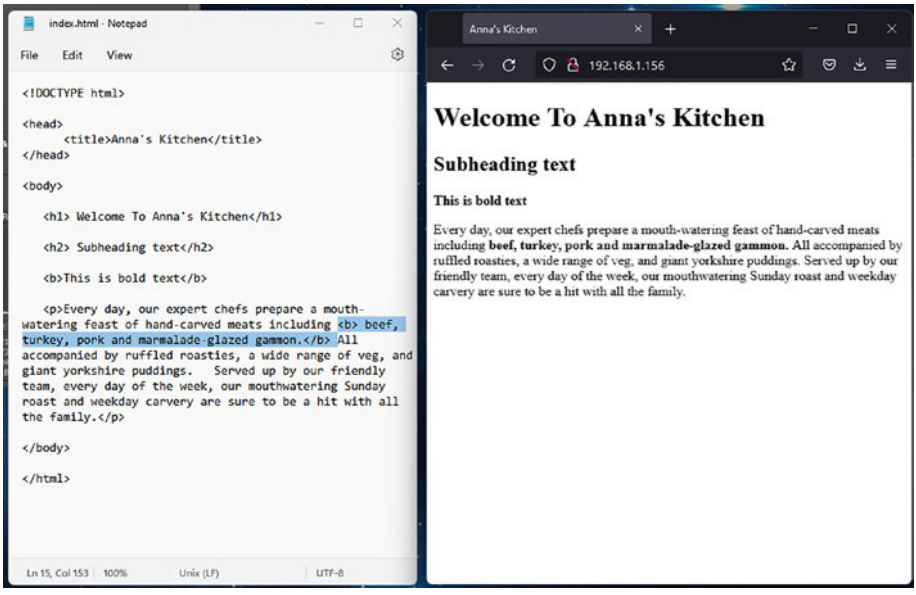


Рис. 3-7. *Жирный текст*

Текст “**говядина, индейка, свинина и окорок, глазированный мармеладом.**” отображается жирным или насыщенным текстом.

Курсивный текст

Вы также можете сделать текст курсивным или подчеркнутым. Тег обозначает альтернативный текст, а содержимое внутри обычно отображается курсивом. Тег помечает текст как подчеркнутый, а содержимое внутри обычно отображается курсивом.

Чтобы использовать эти теги, просто окружите слово или слова тегами или :

**** Наше аппетитное воскресное жаркое и мясные блюда в будние дни подаются нашей дружной командой каждый день недели и обязательно понравятся всей семье. ****

**** Наше аппетитное воскресное жаркое и мясные блюда в будние дни подаются нашей дружной командой каждый день недели и обязательно понравятся всей семье. ****

Посмотрите на пример (Рис. 3-8). Текст "*Наше аппетитное воскресное жаркое и мясные блюда, которые подаются нашей дружной командой каждый день в неделю, обязательно понравится всей семье*" выделен курсивом или подчеркнут.

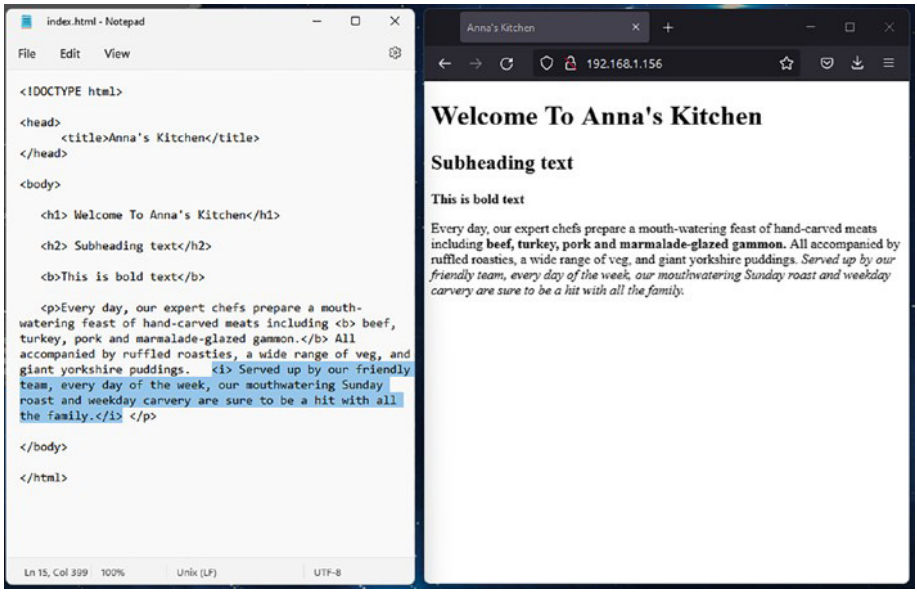


Рис. 3-8. *Курсив*

Теперь давайте соберем все это вместе, используя информацию, которую мы только что узнали. В тело вашего HTML-документа попробуйте добавить пример, показанный на Рис. 3-9.

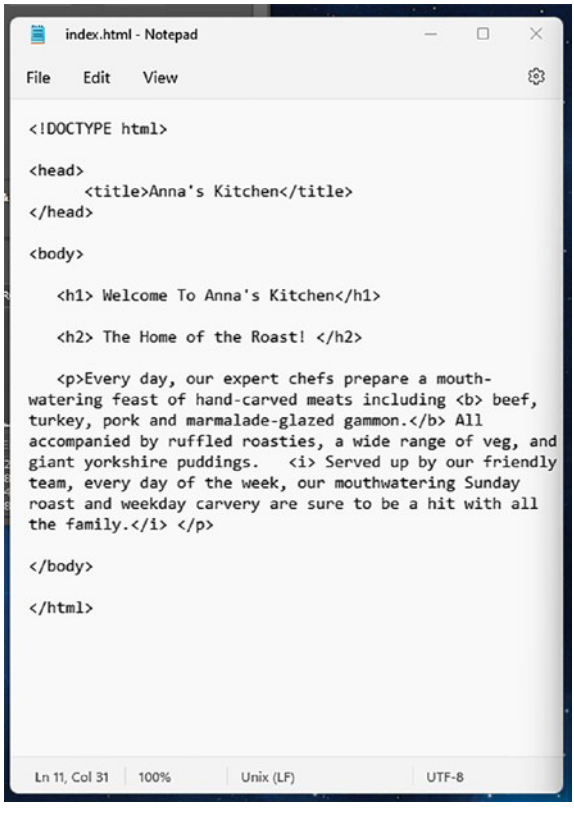


Рис. 3-9. *Использование тегов*

Когда вы откроете страницу в веб-браузере, вы увидите что-то вроде Рис. 3-10.

Рис. 3-10. *Используемые теги*

Заголовок был отформатирован с помощью <H1>, а строка тега была выделена жирным шрифтом с помощью <b>.

Кроме того, мы создали абзац, используя теги <p>. Эти теги разбивают текст на аккуратно расположенные абзацы.

Цвет фона страницы

Чтобы изменить цвет фона любого объекта, добавьте атрибут style. Установите для него значение «background-color», а затем выберите цвет из списка цветов HTML:

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

Взгляните на код на Рис. 3-11.

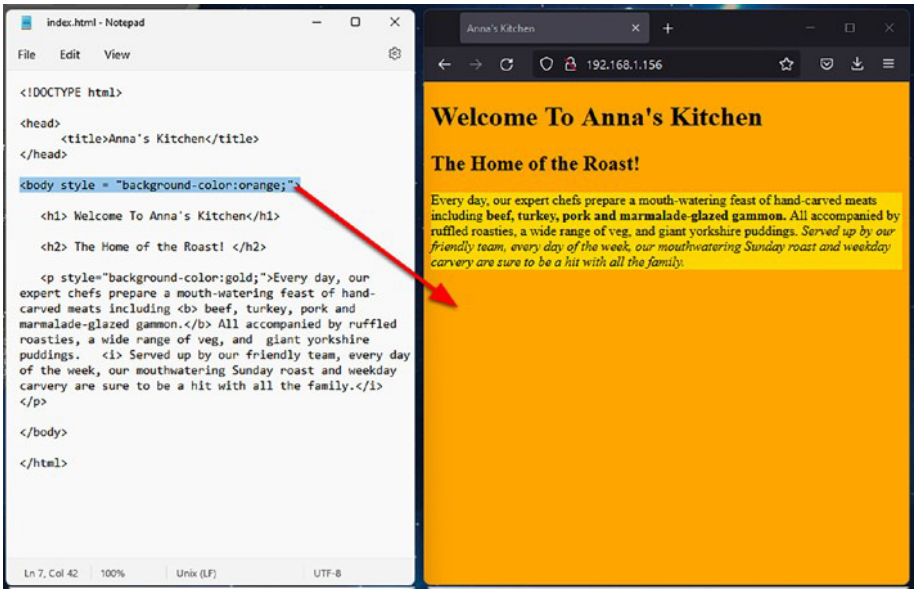


Рис. 3-11. *Цвет фона*

Вы также можете изменить цвет фона других объектов, например фона абзаца (Рис. 3-12):

```
<p style = "background-color:Gold">
```

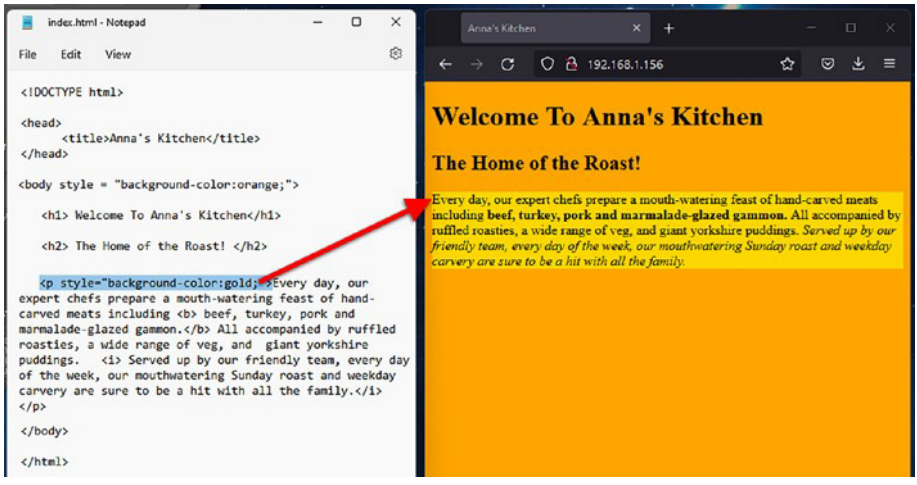


Рис. 3-12. *Цвет фона объектов*

Цвет текста

Чтобы изменить цвет текста, добавьте к стилю атрибут style (Рис. 3-13). Установите атрибут «color», а затем выберите цвет из списка цветов HTML:

<H1 style = "color:Yellow;">

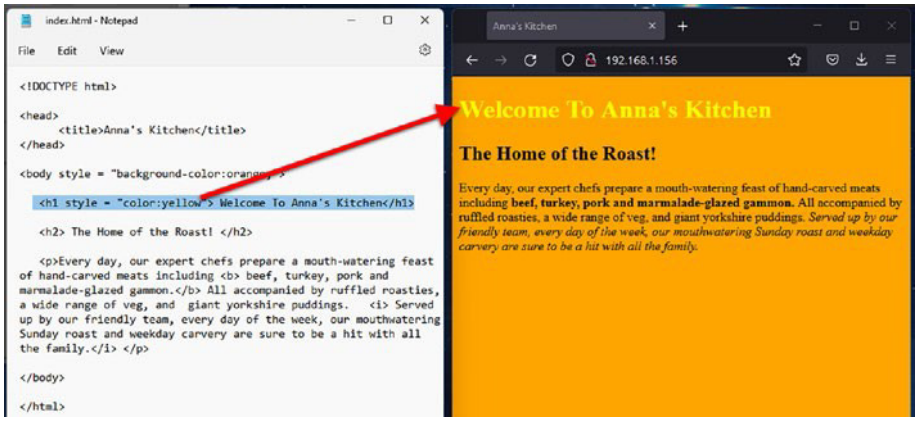


Рис. 3-13. *Изменение цвета текста*

Шрифты

Чтобы изменить шрифт, добавьте к стилю атрибут style. Присвойте атрибуту font-family нужное имя шрифта (Рис. 3-14). В этом примере я использую Helvetica:

```
<h2 style = "font-family:Helvetica;">The Home of the Roast!</h2>
```

Давайте добавим эту строку в наш HTML-файл.

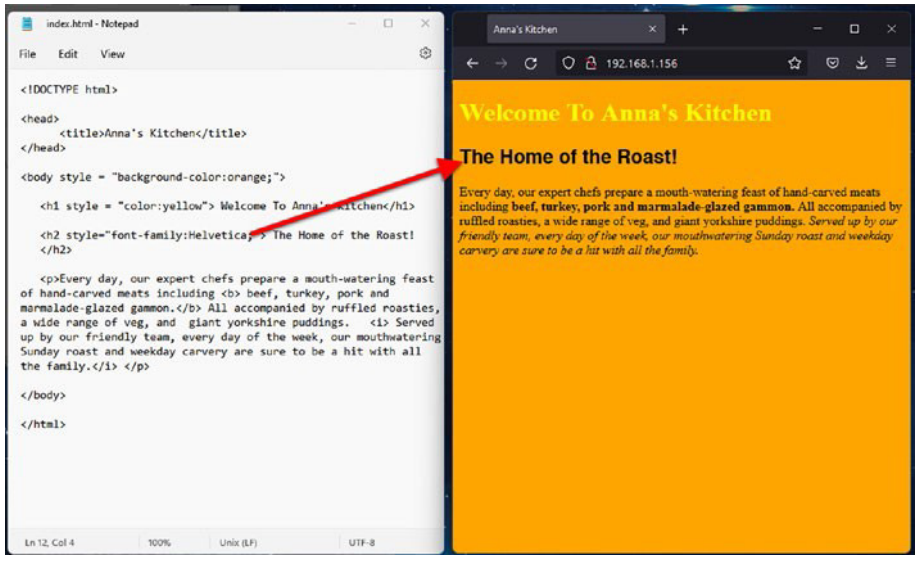


Рис. 3-14. *Изменение шрифта*

Шрифт в подзаголовке изменен на Helvetica.

Вы можете выбирать из большого количества шрифтов. Не все из них поддерживаются всеми браузерами, но большинство из них поддерживаются.

Вы также можете использовать Google Fonts.

HTML-объекты

Объект HTML — это фрагмент текста, который начинается с

амперсанда и заканчивается точкой с запятой и часто используется для отображения зарезервированных символов, которые в противном случае интерпретировались бы как код HTML, невидимых символов, таких как неразрывные пробелы, и символов. Часто используемым объектом в HTML является неразрывный пробел:

Если вы хотите добавить знак авторского права, используйте

©

или, возможно, знак доллара ($) или фунта (£):

$
£

На рис. 3-15 показан список часто используемых символов.

Character	Description	HTML Entity
	non-breaking space	
<	less than	<
>	greater than	>
&	ampersand	&
"	double quotation mark	"
'	single quotation mark (apostrophe)	'
¢	cent	¢
£	pound	£
¥	yen	¥
€	euro	€
©	copyright	©
®	registered trademark	®

Рис. 3-15. *Часто используемые символы*

Добавление изображений

Чтобы добавить изображение, используйте элемент <img>:

<img src = "img/carvery.jpg" width=" " height=" ">

Используйте атрибут src, чтобы указать имя файла и расположение изображения. Рекомендуется хранить изображения в отдельной папке под названием “img,” “images” или иногда “assets”. Это поможет держать все ваши файлы в порядке. В этом руководстве мы сохраняем наши изображения в папке «img» на нашем веб-сервере.

Чтобы сослаться на изображение в атрибуте src элемента img, добавьте имя папки, затем косую черту, а затем имя изображения (Рис. 3-16).

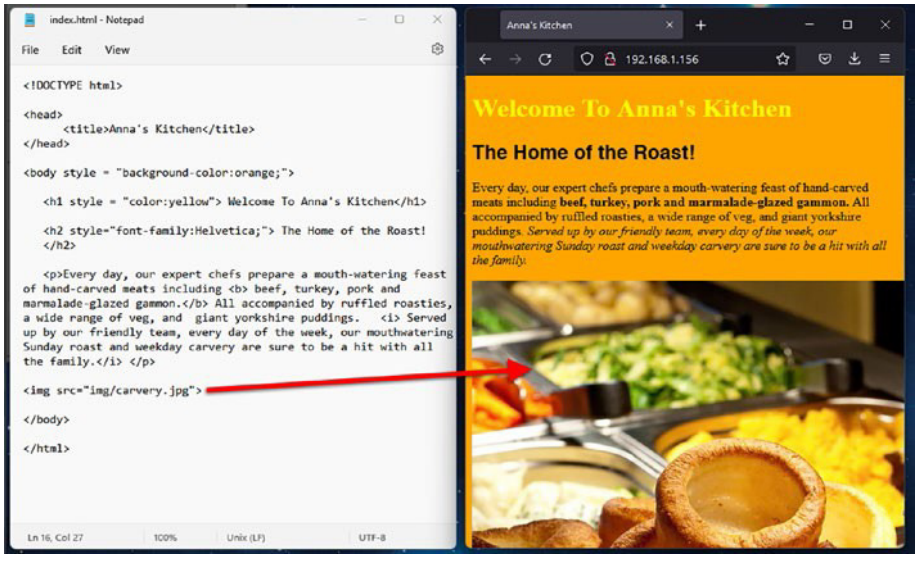


Рис. 3-16. *Добавление изображения*

Обратите внимание, что размер этого изображения большой. Вы также можете указать размер изображения, используя атрибуты ширины и высоты, измеряемые в пикселях (px).

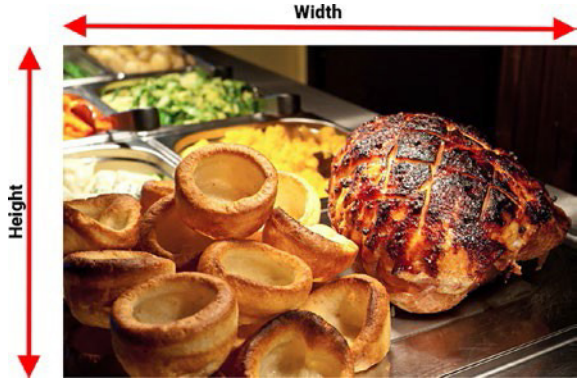


Рис. 3-17. *Размер изображения*

По умолчанию изображение будет отображаться в соответствии с сохраненной шириной и высотой фактического изображения (Рис. 3-17), но вы можете изменить их.

Понимание размеров изображения

Размеры изображения (т.е. ширина и высота) указываются в пикселях (px). Изображение на Рис. 3-18 имеет размер 500 на 220 пикселей. Это означает, что имеется 500 пикселей по ширине и 220 пикселей по высоте.

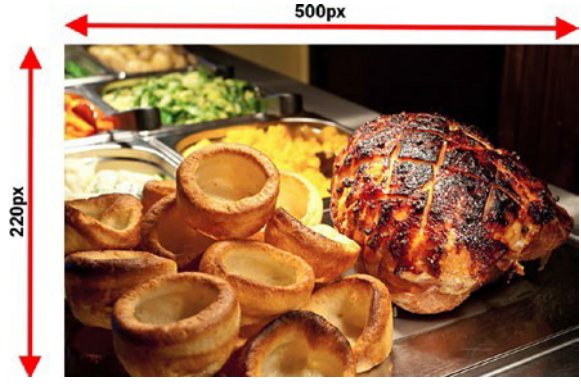


Рис. 3-18. *Изменение размера изображения*

Если вы посмотрите на изображение на Рис. 3-18, оно немного мало и его можно было бы расширить до ширины страницы.

Ширина в 500 пикселей подойдет лучше, поэтому добавьте атрибут ширины и присвойте ему значение «500»(Рис.3-19):

```
<img src="img/carvery.jpg" width="500">
```

Это расширит изображение и автоматически отрегулирует его длину, чтобы предотвратить искажение изображения.

Вы также можете указать ширину и высоту.

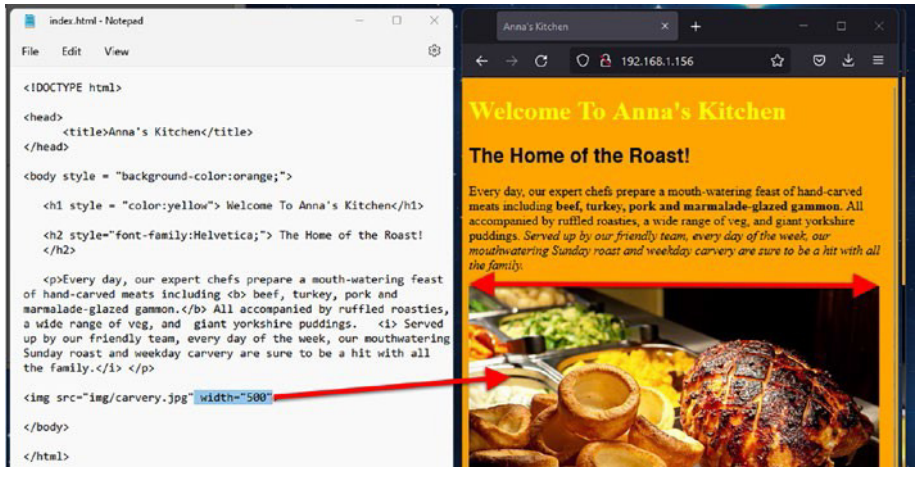


Рис. 3-19. *Настройка размера изображения*

Выравнивание изображения

Изображения можно выравнивать по левому или правому краю страницы, как и абзацы текста, с помощью атрибута выравнивания.

В предыдущем примере, когда мы добавляли изображение на веб-страницу, мы просто добавляли его в нижнюю часть страницы. Изображение по умолчанию выравнивается по левому краю страницы.

Вы можете выровнять изображения на странице по абзацам. Это

делает текст более приятной для чтения.

Для этого вам нужно будет разместить элемент <img> в теге абзаца <p>. Это проще, чем кажется.

Я собираюсь выровнять изображение по правой стороне с помощью атрибута align и поместить элемент <img> после тега <p> абзаца. Взгляните на картинку на Рис. 3-20.

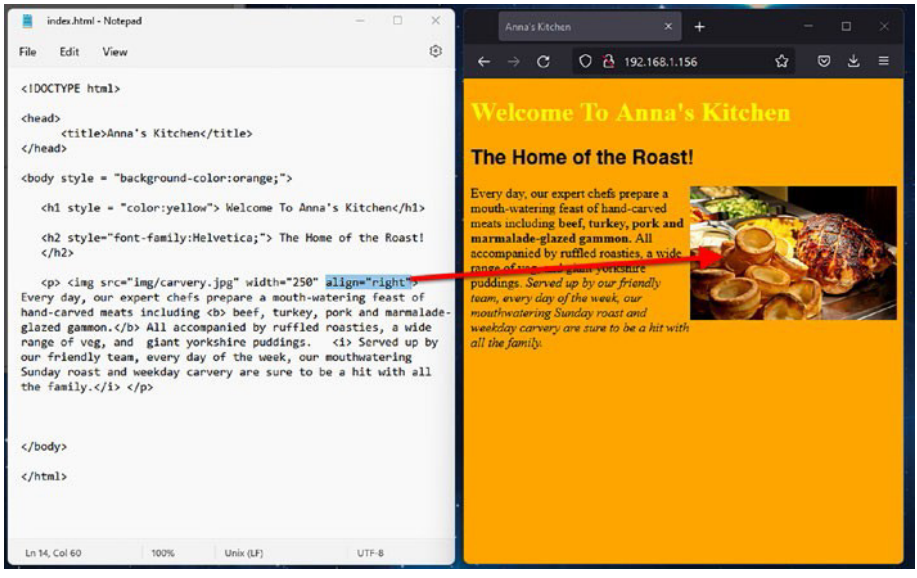


Рис. 3-20. *Выравнивание изображения*

Кроме того, чтобы изображение подошло по размеру, вам нужно изменить его размер, используя атрибут ширины вашего элемента <img>. Ширина нашей страницы составляет 540 пикселей, поэтому примерно половины этой длины будет достаточно. Установите для атрибута ширины значение. Посмотрите на ранее выделенную строку в документе в Блокноте.

Что произойдет, если вы измените атрибут align на «left» или «middle»?

Фоновое изображение

Вы можете добавлять изображения в качестве фона многих элементов HTML, таких как элемент абзаца или таблица. Например, если я хочу добавить фоновое изображение в тело документа:

```
<body style = "background-image: url('img/img-bg.jpg')";
```

На Рис. 3-21 я выделил место, где мы добавили строку.

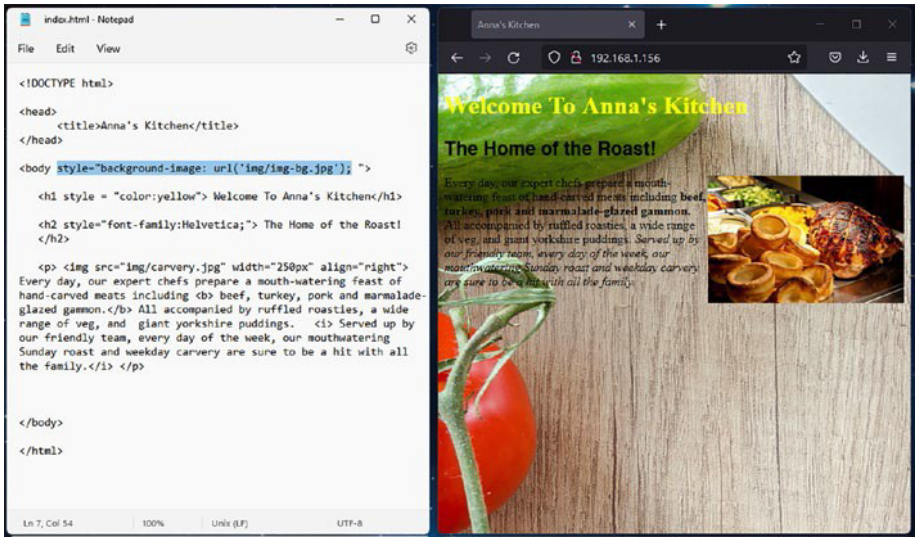


Рис. 3-21. *Изображение добавлено в качестве фона*

Обратите внимание, что изображение слишком велико для экрана.

Мы можем изменить размер ширины, используя атрибут background-size. Первый параметр — ширина, второй — высота:

```
background-size: ширина высота
```

Например:

```
background-size: 100%;
```

Вы также заметите, что фоновое изображение повторяется вниз по странице (Рис. 3-22).

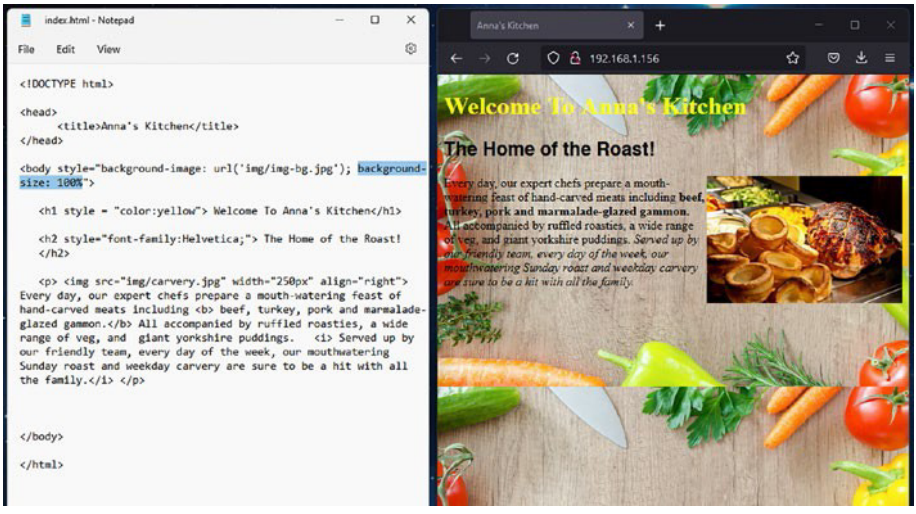


Рис. 3-22. *Изображение повторяется в качестве фона*

Чтобы изменить это, добавьте

background-repeat: no-repeat

Давайте посмотрим. Теперь вы увидите, что фоновое изображение появляется только один раз (Рис. 3-23).

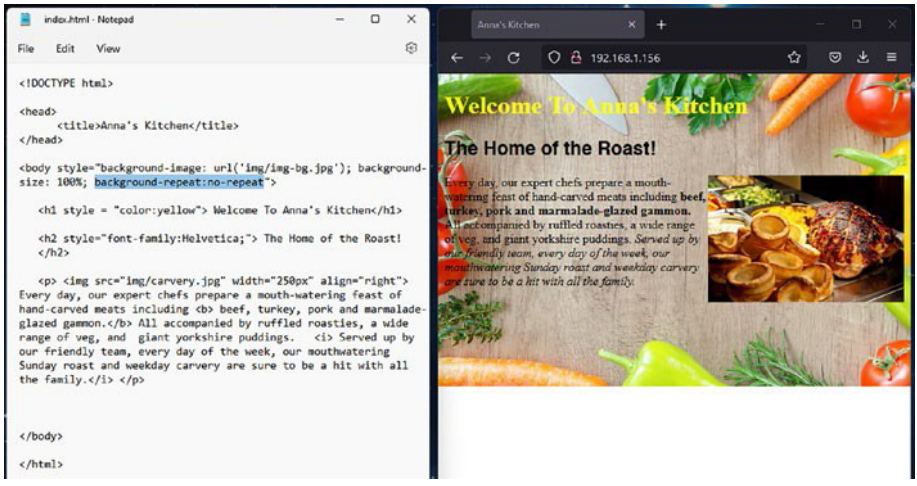


Рис. 3-23. *Фоновое изображение появляется один раз*

Вы также можете добавлять фоновые изображения к другим элементам, например к абзацу:

<p style="background-image: url('img/img_bg.jpg);">

Добавление таблиц

Для создания таблицы используйте следующие теги:

<table> </table>

Внутри этих тегов вы можете использовать следующие теги для определения каждой строки таблицы:

<tr>…</tr>

Используйте следующие теги, чтобы определить каждую запись в этой строке — они станут столбцами:

<td>…</td>

Давайте рассмотрим пример (Рис. 3-24).

Рис. 3-24. *Создание таблицы*

Давайте добавим код на нашу веб-страницу. Введите код таблицы, как показано в Блокноте слева внизу (Рис. 3-25).

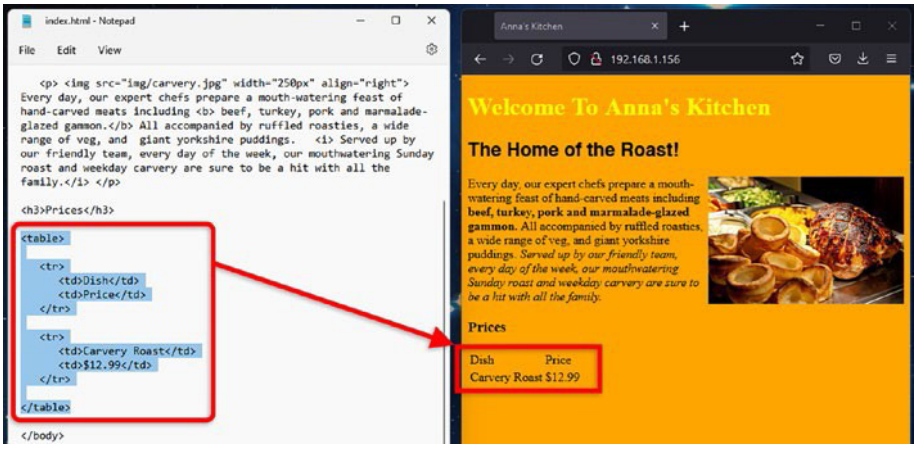


Рис. 3-25. *Таблица на веб-странице*

Добавление ссылок

Ссылки на ваш сайт можно добавлять с помощью тегов привязки <a>.... Вы можете разместить ссылку на другой веб-сайт, другую страницу, документ или обект загрузки.

Начните с тега привязки, затем используйте атрибут href, чтобы указать URL-адрес веб-сайта или страницы, на которую вы хотите создать ссылку. Введите URL-адрес или названи страницы между двоными кавычками:

Добавьте название ссылки, которая будет отображаться на сайте между тегами:

View our Menu

Итак, сложив все это вместе, мы получим вот это:

 View our Menu

Попробуйте добавить строку внизу документа в Блокноте. Мы добавили тег привязки между тегом абзаца, чтобы разделить строки (Рис. 3-26).

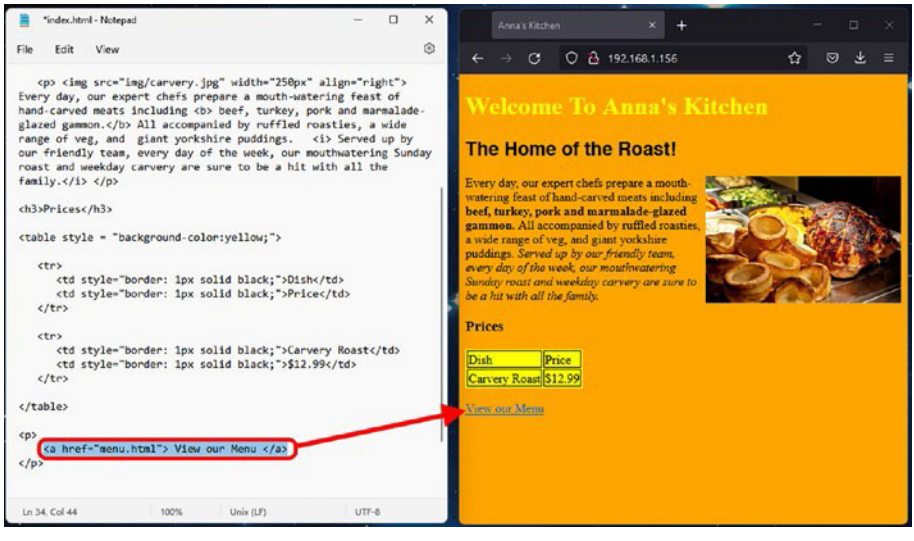


Рис. 3-26. *Связывание меню*

Вы можете видеть, что ссылка на веб-странице справа имеет синий подчеркнутый текст. Это указывает на ссылку.

Вы также можете ссылаться на определенные файлы, такие как изображения, документы или объекты загрузки.

Например, если у вас есть документ в папке «uploads» в каталоге PUBLIC_HTML или htdocs, это может быть документ PDF в папке «uploads» (Рис. 3-27). На нашем личном веб-сервере с помощью проводника вы можете создать новую папку C:\Abyss Web Server\htdocs\.

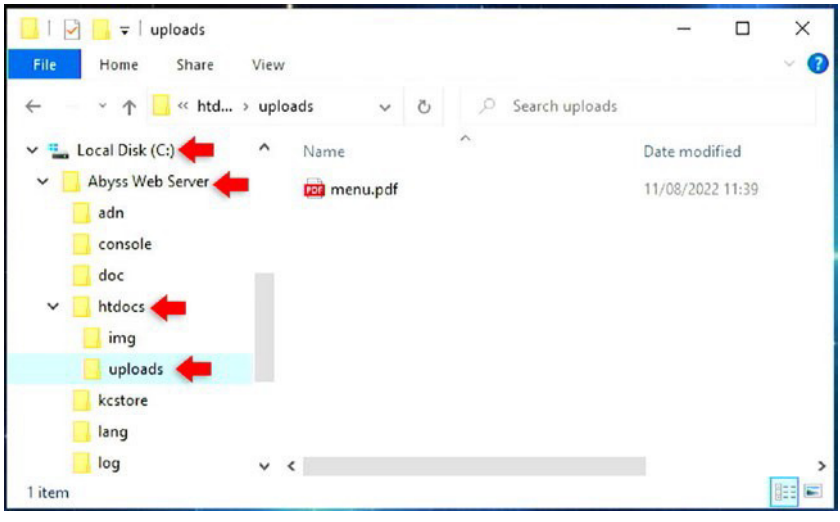


Рис. 3-27. *Ссылка на изображения и документы*

Чтобы создать ссылку на этот файл, мы добавляем наименование каталога перед наименованием документа, например:

```
<a href="uploads/menu.pdf"> Download Now </a>
```

Если ресурс, изображение или страница находится где-то еще в Интернете, вам необходимо добавить полный адрес в атрибут href, например:

www.ellumitechacademy.com/uploads/menu.pdf

или на нашем сервере:

localhost/uploads/menu.pdf

Для 100% совместимости при ссылках на другие сайты и ресурсы за пределами вашего сайта рекомендуется добавить протокол в начало URL-адреса href, например:

https://www.ellumitechacademy.com/uploads/menu.pdf

или на нашем сервере:

http://localhost/uploads/menu.pdf

Другими протоколами могут быть

ftp:
mailto:
file:
http:
https:

в зависимости от того, где размещен ваш ресурс.

Использование изображений в качестве ссылок

Вы также можете превратить в ссылку изображение. Для этого все, что вам нужно сделать, это вставить свое изображение:

<img src = "carvery.jpg">

между тегами привязки <a>.... Я хочу, чтобы изображение ссылалось на домашнюю страницу.

Итак, вы получаете что-то вроде этого:

 <img src="carvery.jpg">

Давайте посмотрим; добавьте код ниже. Я использовал изображение carvery.jpg из каталога изображений, поэтому обязательно добавьте

images/

перед именем файла в атрибуте src тега <img>. Посмотрите на код, выделенный в документе «Блокнот» слева (Рис. 3-28).

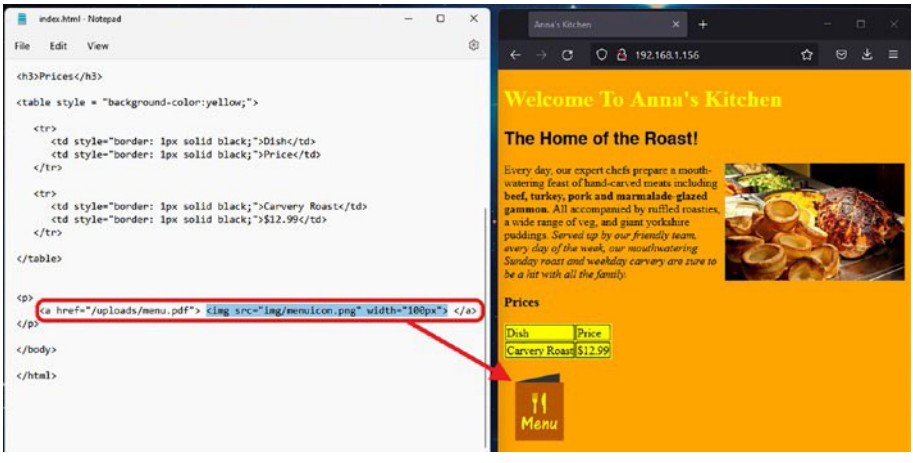


Рис. 3-28. *Ссылка на меню в вид изображения*

Обратите внимание, что указатель мыши превращается в руку, когда вы наводите курсор на изображение (Рис. 3-29). Это указывает на ссылку.

Рис. 3-29. *Появляется ссылка на меню*

Обычно целевой URL-адрес можно увидеть в строке состояния в нижней части веб-браузера.

Сохранение форматирования

Иногда вам может потребоваться, чтобы ваш текст точно соответствовал тому формату, в котором он написан в документе HTML. В этих случаях вы можете использовать теги предварительного форматирования:

<pre>... </pre>

Добавление списков

Неупорядоченные списки отображаются в виде маркированных списков. Упорядоченные списки отображаются как нумерованные списки.

Неупорядоченный список

Используйте теги <ul>.... Для каждого элемента списка вам нужно будет добавить текст между тегами <li>... (Рис. 3-30).

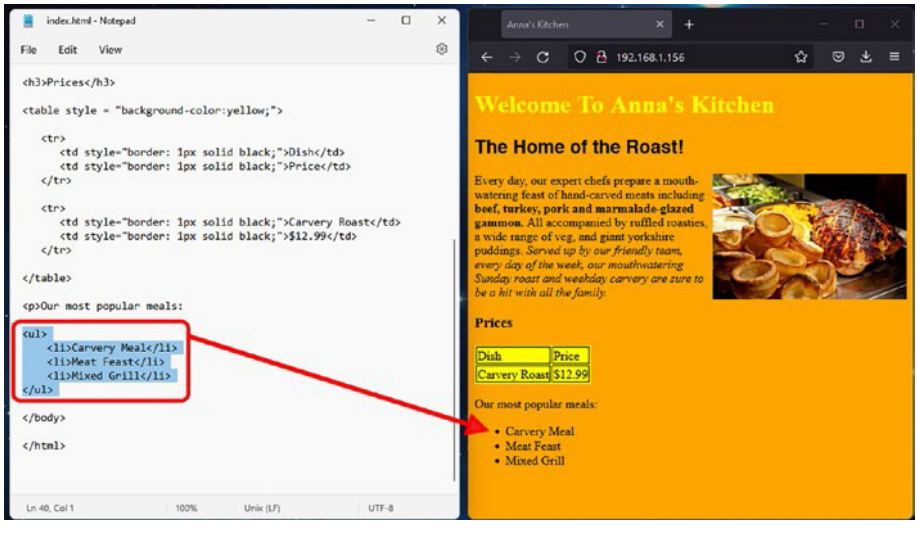


Рис. 3-30. *Неупорядоченный список*

Упорядоченный список

Используйте теги <ol>.... Для каждого элемента списка вам нужно будет добавить текст между тегами <li>... (Рис. 3-31).

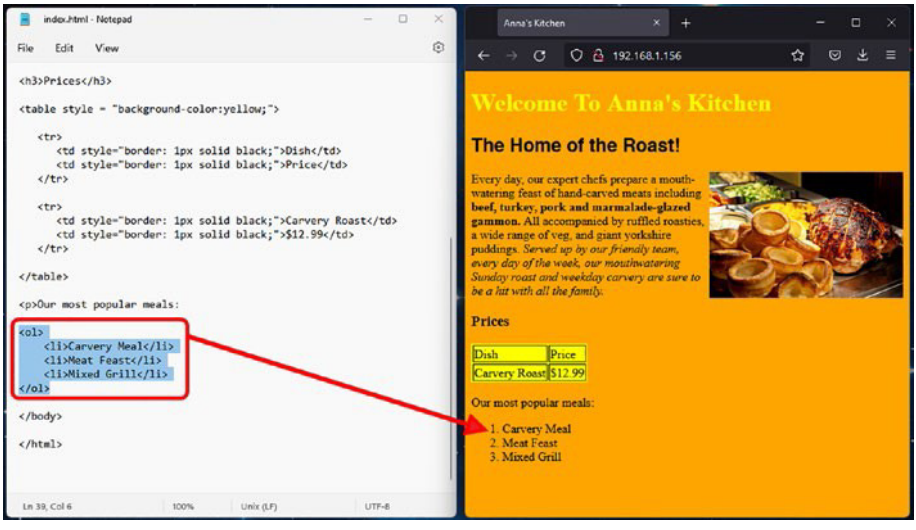


Рис. 3-31. *Упорядоченный список*

Структурирование вашей веб-страницы

HTML5 представил несколько новых тегов макета и структурирования, которые помогают нам определять и форматировать нашу веб-страницу. Они известны как семантические элементы и могут использоваться для определения различных частей веб-страницы. Семантические элементы четко описывают значение контента для браузера и разработчика. Другими словами, элементы имеют осмысленные наименования.

Эти теги полезны для пользователей, которые полагаются на программы чтения с экрана. Такие программы чтения озвучивают содержимое страницы, и если вы используете семантические

элементы, это позволяет программам чтения с экрана более точно передавать содержимое страницы.

Поисковые системы, такие как Google и Bing, будут использовать семантические элементы, чтобы идентифицировать и выяснить, какие части вашего сайта содержат наиболее важный контент.

В Таблице 3-1 перечислены несколько общих элементов.

Таблица 3-1. *Общие элементы*

<header> … </header>	контейнер для вводного контента, заголовков страниц или заголовков
<nav> … </nav>	раздел страницы, используемый для размещения навигационных ссылок, например меню сайта
<main> … </main>	содержит основной контент страницы
<section> … </section>	отдельный раздел главной страницы
<article> … </article>	автономный раздел в документе или странице, который можно использовать повторно, например, в статье блога или виджете
<aside> … </aside>	позволяет вам определить некоторый контент помимо основного контента, например боковую панель
<footer> … </footer>	содержит такую информацию, как сноски, данные об авторе и авторских правах
<figure> … </figure>	содержит фотографии, изображения, иллюстрации или диаграммы
<figcaption> … </figcaption>	позволяет вам определить заголовок для элемента <figure>
<Заключение> … </Заключение>	позволяет определить видимый заголовок для элемент <details>

<details> … </details>	позволяет указать контент, который пользователь может открывать и закрывать
…	определяет раздел или раздел в документе HTML и может использоваться в качестве контейнера для элементов HTML, стилизованных с помощью CSS или управляемых с помощью атрибута class или id. См. Главу 4

Давайте начнем добавлять их в соответствующие разделы нашей HTML-страницы. На нашей странице есть заголовок «Welcome to Anna's Kitchen», поэтому мы можем окружить его тегами <header> и панель навигации, чтобы мы могли окружить карту навигационного изображения тегами <nav> (Рис. 3-32).

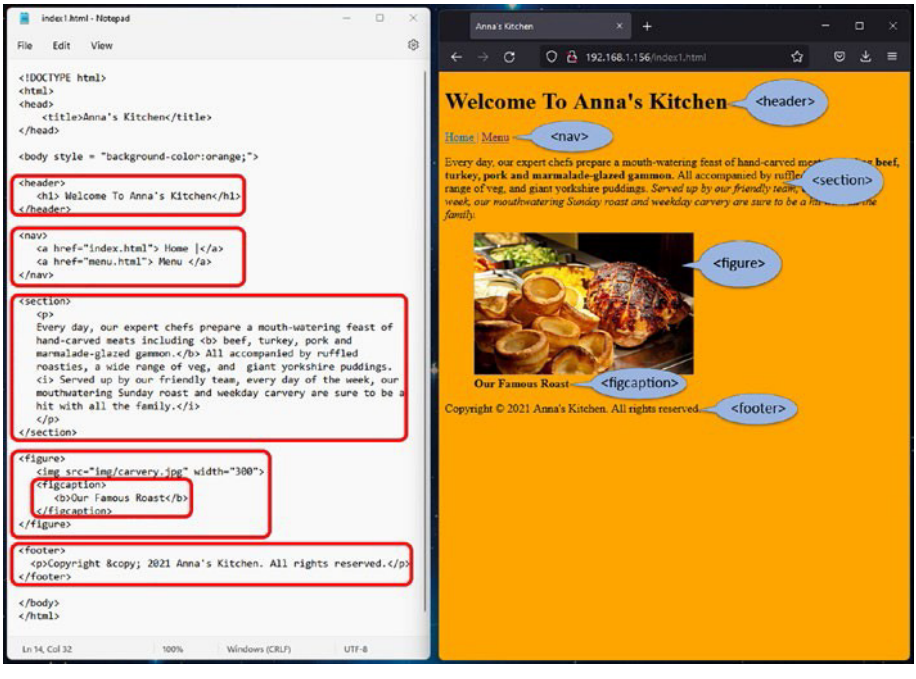


Рис. 3-32. *Теги*

Основной контент может размещаться в теге <section>, любые изображения могут размещаться на вкладке <figure>, а нижний колонтитул можно добавить с помощью тегов <footer>.

Упражнение

1. Откройте новый текстовый файл и сохраните его как ch03.html.
2. Напишите базовую структуру HTML-документа.
3. Как написать жирный текст, используя элементы HTML?
4. Как определить тело документа?
5. Как создавать заголовки с помощью элементов HTML?
6. В HTML-документе ch03.html создайте заголовок.
7. Измените цвет фона на #E2E0E2 или на любой другой по вашему выбору.
8. Начните новый абзац, затем добавьте абзац текста.
9. Вставьте гиперссылку на документ, созданный вами в предыдущей главе (ch02.html).

Заключение

- При использовании локального сервера Abyss на вашем компьютере сохраняйте документы в **C:\Abyss Web Server\htdocs**.
- Для доступа к веб-сайту на вашем компьютере с помощью браузера используйте **http://127.0.0.1**.
- Стиль основного заголовка: **<h1>...</h1>**
- Стиль подзаголовка: **<h2>...</h2>**

- Стиль незначительного подзаголовка: **<h3>...</h3>**
- Жирный текст: **...**
- Курсивный текст: **...**
- Текст абзаца: ** ... **
- Добавьте атрибут стиля для изменения цвета, например, **<H1 style = “color:Yellow;”>**.
- Добавьте атрибут стиля, чтобы изменить шрифт, например, **<H1 style = “font-family:Helvetica;”>**.
- To add a table, use **<table> </table>**.
- To define each table row, use **<tr>...</tr>**.
- To define data to that row, use **<td>...</td>**.
- Use the img element to add an image: **<img src = “ ” width = “ ”>**.
- Use the anchor element to define hyperlinks: **<a href = “ ”>**.

ГЛАВА 4

Каскадные таблицы стилей

Каскадные таблицы стилей (CSS) используются для определения и настройки стилей и макетов ваших веб-страниц. Это означает, что вы можете создавать таблицы стилей для изменения дизайна, макета и реагирования на различные размеры экрана на различных устройствах, от компьютеров до смартфонов.

CSS описывает, как элементы HTML должны отображаться на экране, и управляет макетом нескольких веб-страниц одновременно. Это связано с тем, что таблицы стилей хранятся в отдельных файлах CSS и связаны с документом HTML.

CSS решил большую проблему. Изначально HTML никогда не предназначался для содержания тегов для форматирования веб-страницы и был создан для описания содержимого указанной веб-страницы. Когда в спецификацию HTML 3.2 было добавлено больше атрибутов форматирования, для веб-разработчиков стало настоящим кошмаром проектирование и поддержка веб-сайтов. Это связано с тем, что информация о шрифтах, форматировании, макете и цвете добавлялась в каждый HTML-тег на каждой странице, поэтому внесение изменений и поддержание веб-сайта было долгим и дорогостоящим процессом.

Чтобы решить эту проблему, Консорциум Всемирной паутины (W3C) создал и внедрил CSS. CSS исключил форматирование стиля из HTML-страницы и позволил разработчику включать информацию о

форматировании и макете в отдельный файл, который можно было включить во все остальные HTML-страницы, составляющие веб-сайт.

Слово «каскадирование» означает, что правила стилизации исходят из нескольких источников. Это означает, что CSS имеет иерархию, и стили с более высоким приоритетом перезапишут стили с более низким приоритетом. Другими словами, стили, расположенные ниже по иерархии, имеют более высокий приоритет по сравнению со стилями, расположенными выше.

Существует три метода, которые вы можете использовать для включения стилей CSS в ваш HTML-документ.

Во-первых, вы можете включить их в текст, используя атрибут style в открывающем теге:

```
<h1 style="color:blue; font-size:14px;"> Heading 1</h1>
```

Вы можете встроить стили, используя элемент <style> в заголовке документа:

```
<head>
    <style>
        H1 {
            color:blue;
            font-size:14px;"
        }
    </style>
</head>
```

Вы можете включить стили CSS, сохраненные в другом файле, используя элемент <link> с атрибутом href, указывающим на файл CSS:

```
<link rel="stylesheet" href="style.css">
```

Внешние CSS-файлы

Рекомендуется добавить объявления CSS в отдельный текстовый файл и связать этот файл со всеми соответствующими HTML-файлами (Рис. 4-1). Таким образом, вы сохраните все объявления стилей в одном месте и сможете легко что-то изменить.

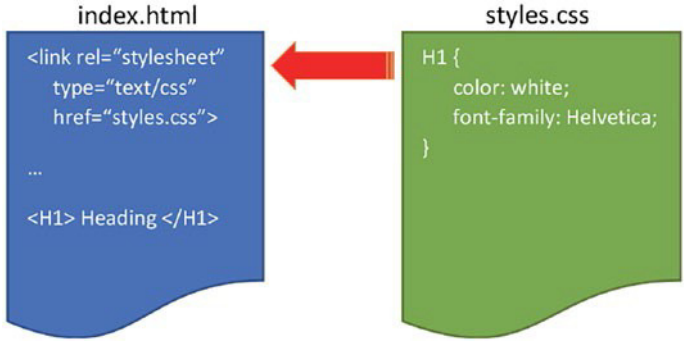


Рис. 4-1. *Связывание файла CSS с HTML-страницей*

Как мы видели в предыдущей главе, нам приходилось стилизовать каждый элемент каждый раз, когда мы его использовали. Это называется встроенными стилями, которые очень неэффективны. Что произойдет, если у нас большой веб-сайт, и мы оформили каждый заголовок белым цветом размером 20 пикселей, используя шрифт Helvetica, а клиент хочет изменить цвет текста или шрифта. Нам придется просмотреть каждый экземпляр и изменить его. Для меня это звучит как кошмар.

Гораздо лучший способ — определить все элементы, теги и т.д. с помощью таблицы стилей. Именно здесь CSS показывает свою истинную силу. Если бы клиент пришел с предыдущим запросом и мы использовали таблицы стилей CSS, нам нужно было бы только изменить объявление в CSS, и каждый экземпляр изменился бы по всему сайту автоматически.

Создайте текстовый файл с расширением .CSS и убедитесь, что он находится в том же каталоге, что и ваши HTML-файлы (Рис. 4-2).

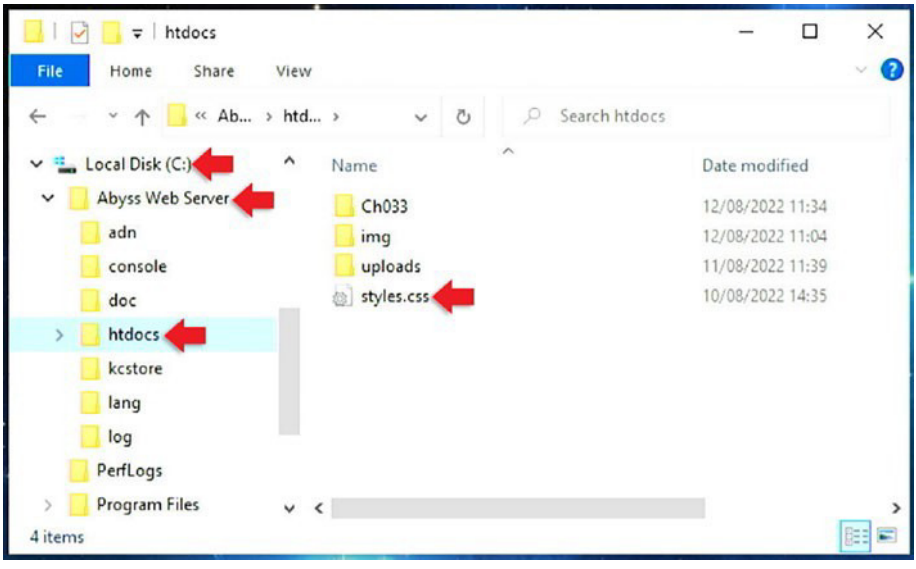


Рис. 4-2. *Текстовый файл .CSS*

Add this line in the <head> section of each HTML file that is to be styled using the declarations contained in the CSS file. Use the href attribute to point to the CSS file:

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

CSS-синтаксис

Давайте посмотрим на базовый синтаксис правила CSS. Как вы можете видеть на Рис. 4-3, правило CSS состоит из селектора и блока объявлений.

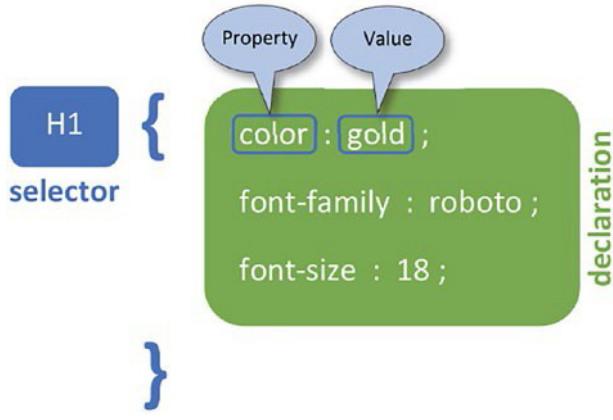


Рис. 4-3. *CSS-синтаксис*

Селектор указывает на HTML или элемент, который вы хотите стилизовать. Блок объявлений начинается с фигурной скобки и содержит одно или несколько объявлений стилей, разделенных точкой с запятой. Каждое объявление включает имя свойство CSS и значение, разделенные двоеточием.

Вы можете использовать их для настройки стилей классов и селекторов, используя различные свойства, как показано на Рис. 4-4.

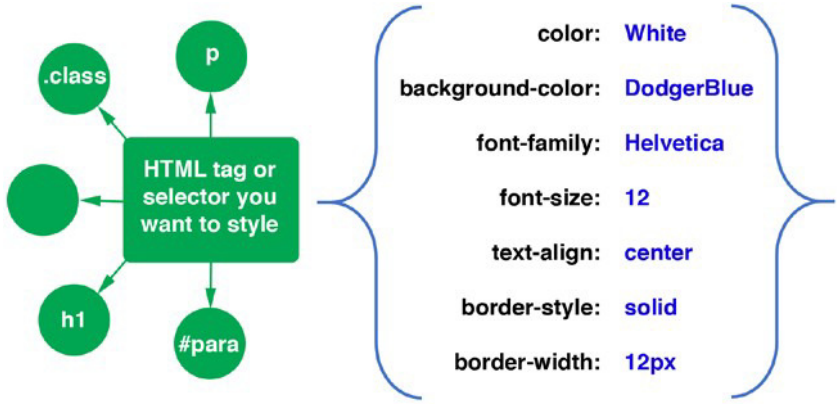


Рис. 4-4. *Стилизация различных селекторов CSS*

Выбор типа элемента

Это позволяет создавать общий стиль для объявленного элемента, и лучше всего его использовать, когда все экземпляры этого элемента должны быть оформлены одинаково:

```
H1 {
    color: white;
    font-family: Helvetica;
}
```

Здесь все элементы H1 будут оформлены белым шрифтом Helvetica.

Селектор классов

Селектор классов используются для применения стилей к определенному элементу HTML. Вы можете назвать класс как угодно, и это название должно начинаться с точки (или разделителя). Используйте селекторы классов, если вы хотите стилизовать несколько элементов на странице или сайте с одинаковым внешним видом или макетом.

Итак, в этом примере я создаю стиль выделения, который можно применить к различным элементам HTML, таким как заголовки <H1> и <H2> или абзац <p>:

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

В своем HTML-коде назначим селектор классов **.highlight**, который мы определили в объявлениях CSS, используя атрибут class в любом элементе HTML. Например, если бы я хотел выделить заголовок в теге <h2>, я бы использовал атрибут class и назначил селектор класса, который я определил ранее:

```
<h2 class = "highlight">
    The Home of the Roast
</h2>
```

Аналогично, если бы я хотел выделить абзац

```
<p class = "highlight">
    Every day, our expert chefs...
</p>
```

Давайте добавим это на нашу небольшую веб-страницу. Я объявил класс **.highlight** в файле styles.css. Вы можете увидеть это выделенным на Рис. 4-5. Я применил селектор класса **.ighlight** к HTML-тегу <h2>, выделенному в файле index.html.

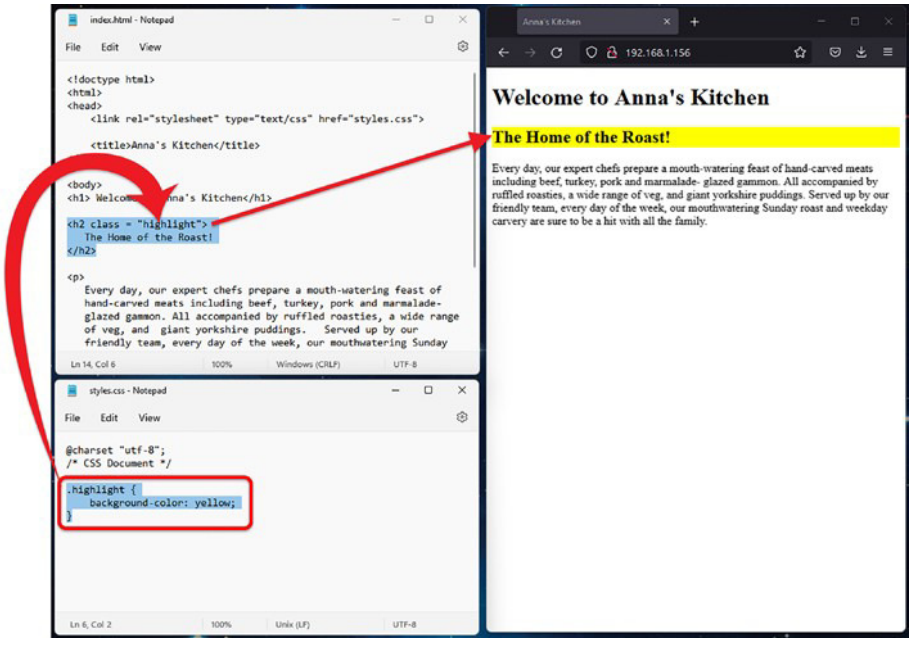


Рис. 4-5. *Класс .highlight*

ID-селектор

ID-селектор нацелен на один элемент и может использоваться только один раз на странице.

ID-селекторы определяются в объявлениях CSS с использованием хэштега # и должны использоваться только в том случае, если на странице есть один элемент, который будет иметь этот конкретный стиль или макет.

Здесь мы собираемся установить золотой цвет фона нижнего колонтитула, и мы хотим, чтобы нижний колонтитул был высотой 100 пикселей, текст выровнен по центру, по горизонтали и вертикали:

```
#footer {
    background-color : gold;
    line-height : 50px;
    text-align : center;
    vertical-align : middle;
}
```

Теперь вы можете использовать идентификатор нижнего колонтитула в теге div, например:

```
<div id= "footer"> </div>
```

Универсальный селектор

Универсальный селектор соответствует каждому элементу на странице. Универсальный селектор полезен, когда вы добавляете определенный стиль ко всем элементам HTML на вашей веб-странице:

```
* {
   margin: 5;
   padding: 5;
}
```

Группировка селекторов

Если вы хотите стилизовать более одного селектора с одинаковыми стилями, вы можете сделать это, сгруппировав селекторы вместе.

Например, если бы я хотел создать стиль, в котором все мои заголовки располагались о центру и были окрашены в золотой цвет, вместо того, чтобы объявлять их все по отдельности, как это показано ниже:

```
h1 {
    text-align: center;
    color: gold;
}

h2 {
    text-align: center;
    color: gold;
}

h3 {
    text-align: center;
    color: gold;
}
```

Я могу сгруппировать все селекторы вместе посредством последующего объявления:

```
h1, h2, h3 {
    text-align: center;
    color: gold;
}
```

Это намного эффективнее и позволит вам объявить стили один раз.

Стилизация текста

Если бы я хотел стилизовать тег H1 для своих заголовков, я мог бы написать в файле styles.css что-то вроде этого:

```
H1 {
    font-color:  yellow;
    font-family: Roboto;
}
```

Это приведет к стилизации всех тегов H1, используемых впоследствии в файле HTML.

Если бы я хотел оформить свои подзаголовки <H2>, я мог бы сделать то же самое. Я хочу изменить цвет шрифта на желтый, а начертание на шрифт Roboto, но на этот раз я хочу сделать текст более тяжелым или жирным. Я могу сделать это, добавив свойство font-weight:

```
H2 {
    color: yellow;
    font-family: Roboto;
    font-weight: 400;
}
```

Давайте посмотрим, что происходит, когда мы добавляем этот код на наш сайт (Рис. 4-6).

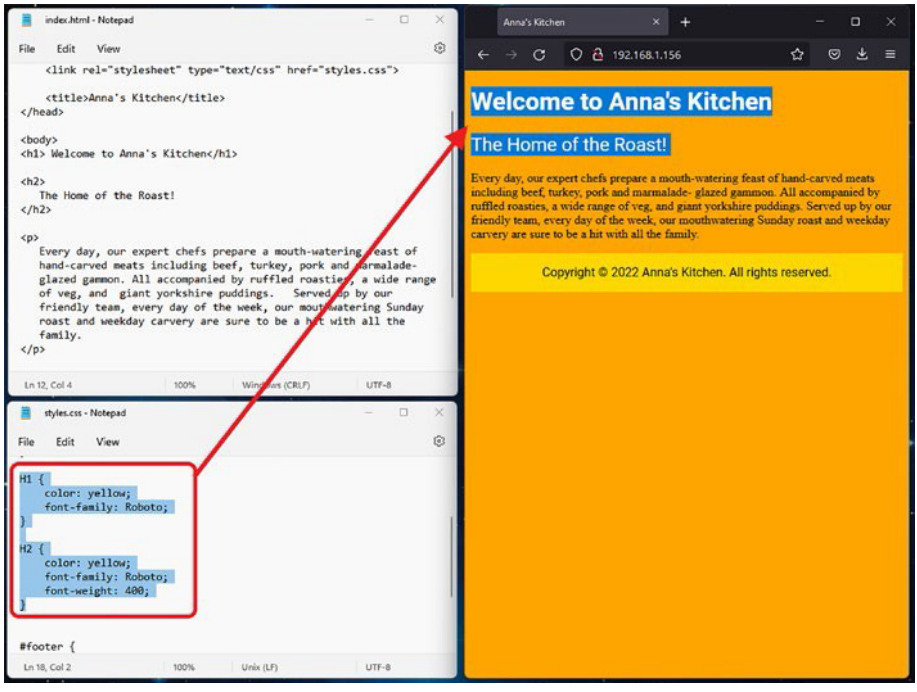


Рис. 4-6. *Добавление цвета на веб-сайт*

Как вы можете видеть на Рис. 4-7, заголовки изменили шрифт и цвет.

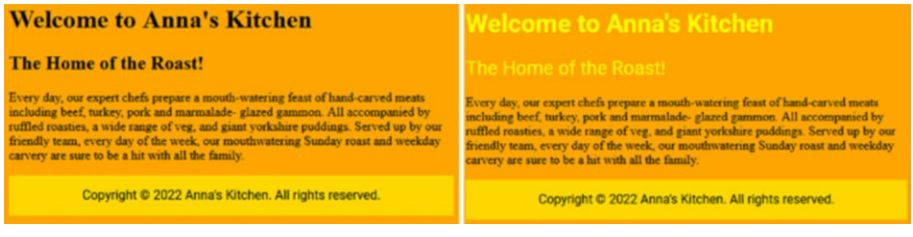


Рис. 4-7. *Изменение цвета шрифта*

Попробуйте изменить цвет и выравнивание шрифта в файле styles.css и посмотрите, что произойдет.

Определение цветов

Вы можете задавать цвета, используя следующие форматы:

- Ключевое слово цвета, например “red”, “green”, “blue”, “transparent”, “orange” и т.д.
- Шестнадцатеричное значение, например “#000000”, “#00A500”, “#FFFFFF” и т.д.
- Значение RGB, например “rgb(255, 255, 0)”

Ключевое слово

Вы можете использовать ключевое слово для желаемого цвета, например черного, белого, темно-синего, серебряного, желтого, оранжевого, темно-оранжевого, золотого и так далее. Полный список см. в Приложении C. Например:

```
h1 {
    color: orange;
}
```

Шестнадцатеричное значение

Шестнадцатеричное значение представляет цвета с помощью шестизначного кода, которому предшествует символ решетки. Код разделен на три двузначных шестнадцатеричных числа, которые представляют количество красного, зеленого и синего цвета различной интенсивности для создания нужного цвета. Значения представлены с использованием шестнадцатеричной схемы нумерации, а не десятичной. Полный список см. в Приложении C.

hex	0	1	2	3	4	5	6	7	8	9	a	B	C	d	e	F
decimal	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Так, например, чтобы создать оранжевый, нам нужно 100% красного, немного зеленого и 0% синего. FF в шестнадцатеричном формате — 255 в десятичном; A5 в шестнадцатеричном формате равно 165 в десятичном формате. Шестнадцатеричный код оранжевого цвета будет выглядеть так

red	green	Blue
FF	a5	00

Мы можем использовать этот шестнадцатеричный код для обозначения желаемого цвета:

```
h1 {
    color: #FFA500;
}
```

RGB-значение

Вы можете указать цвет с помощью функции rgb(). Эта функция принимает три значения от 0 до 255, которые определяют количество красного, зеленого и синего различной интенсивности для создания нужного цвета. Итак, чтобы создать оранжевый, нам нужен красный, смешанный с небольшим количеством зеленого, но без синего.

red	green	Blue
255	165	0

```
h1 {
```

```
    color: rgb(255, 165, 0);
}
```

Полный список цветовых кодов см. в Приложении C.

Понимание единиц измерения

Чтобы контролировать размер определенных объектов, таких как шрифты или изображения, размеры указываются с использованием различных единиц измерения. Существует два типа: абсолютные и относительные.

Абсолютные единицы имеют одинаковые значения независимо от размера родительского элемента или окна. В Таблице 4-1 показаны некоторые примеры.

Таблица 4-1. *Значение единиц измерения*

Единица	Значение
in	дюйм
mm	миллиметр
px	сокращение для пикселей и обычно используется для отображения размеров изображения
pt	сокращение для точек и используется для отображения размера шрифта

Единицы измерения масштаба относительно родительского элемента или размера окна в зависимости от используемой единицы измерения. Вот некоторые примеры:

Единица	Значение
em	относительно текущего размера шрифта элемента. если размер шрифта 12 пт, каждая единица em будет равна 12 пт, поэтому 2em будет 12 x 2, что составляет 24 пт, аналогично 1,5em будет равно 18 пт
%	относительно родительского элемента или размера окна

Отступы, поля и границы

HTML-элементы можно рассматривать как блоки. Блочная модель CSS — это, по сути, блок, охватывающий каждый элемент HTML. Вокруг контента расположены различные слои; первый — это отступы, затем граница, затем поля (Рис. 4-8).

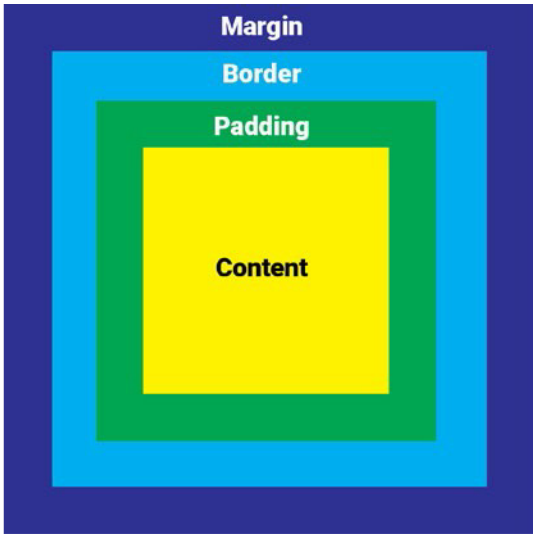


Рис. 4-8. *Блочная модель*

Давайте рассмотрим разные слои. На Рис. 4-8 мы видим, что

- В блоке содержимого отображаются текст и изображения.
- Свойство padding используется для добавления пространства вокруг содержимого внутри определенной границы.
- Свойство margin используется для добавления пространства вокруг содержимого за пределами определенной границы.
- Свойство border-style указывает, какую границу отображать.

Итак, например, давайте добавим стиль к нашему заголовку H1:

```
.myClass {
  padding: 10px;
  border: solid 5px black;
  margin: 20px;
}
```

Когда мы откроем HTML-страницу в веб-браузере, мы увидим что-то вроде Рис. 4-9.

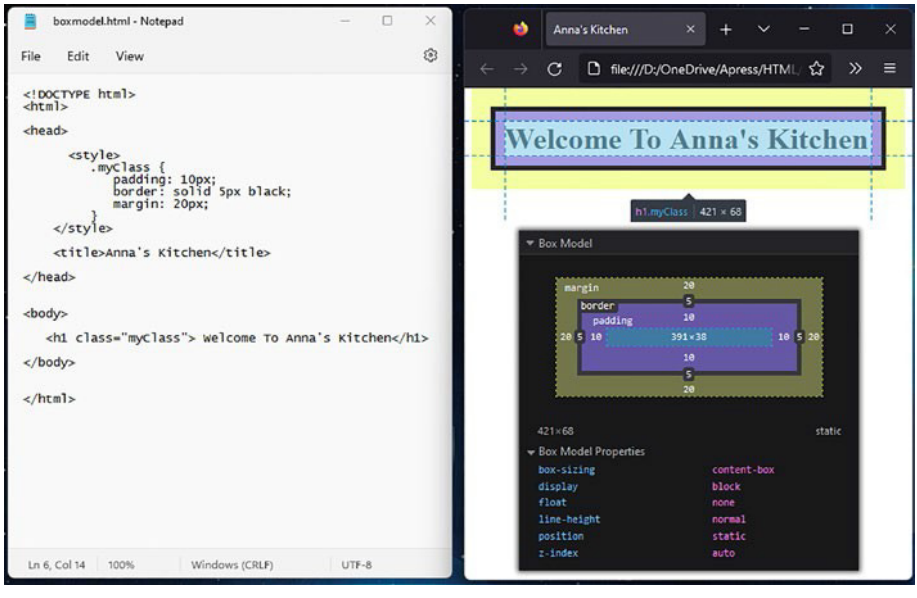


Рис. 4-9. *Блоки в действии*

Если вы посмотрите в веб-браузере, вы увидите блочную модель вокруг заголовка. Пунктирные линии обозначают границы содержимого. Затем мы увидим отступы толщиной 10 пикселей, затем черную рамку толщиной 5 пикселей и поле толщиной 20 пикселей за пределами границы.

Макеты

В предыдущем разделе веб-сайт, над которым мы работали, был очень линейным, то есть каждый раздел просто размещался под следующим. С таблицами стилей вы не ограничиваетесь только изменением стиля HTML-тегов, вы можете также определять макеты и разделы.

Флексбокс

Флексбокс — это модуль макета, предназначенный для размещения контента в одном измерении (строка или столбец, а не оба одновременно) и он лучше всего работает с элементами разных размеров. Флексбокс состоит из флекс-контейнеров (называемых родительскими элементами), которые содержат флекс-элементы (называемые дочерними элементами) (Рис. 4-10).

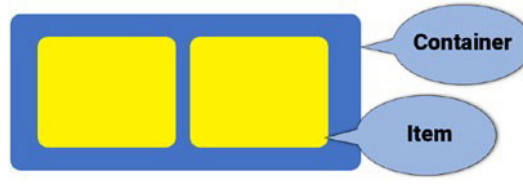


Рис. 4-10. *Контейнер флексбокса*

Элементы флекс-контейнера могут располагаться в любом направлении и изменять свои размеры, что означает, что элементы могут увеличиваться, заполняя неиспользуемое пространство, или

уменьшаться в зависимости от размера экрана.

Мы можем создать контейнер в нашем CSS-коде следующим образом:

```
.flex-container {
  display: flex;
}
```

Свойство flex в элементе отображения определяет, как флекс-элемент будет увеличиваться или уменьшаться, чтобы соответствовать доступному пространству в его контейнере.

Мы можем изменить направление контейнера, чтобы охватить строки или столбцы (Рис. 4-11).

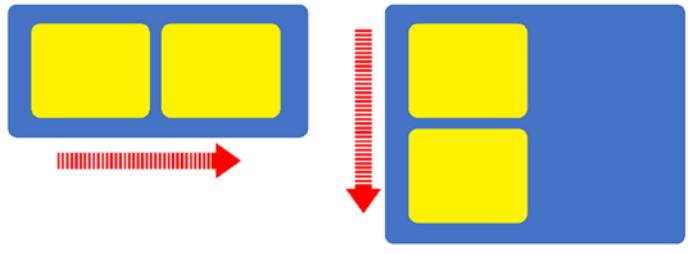


Рис. 4-11. *Контейнер флексбокса, заполняющий строки и столбцы*

Если мы хотим, чтобы элементы были расположены рядом друг с другом в ряд (Рис. 4-12). Это значение по умолчанию.

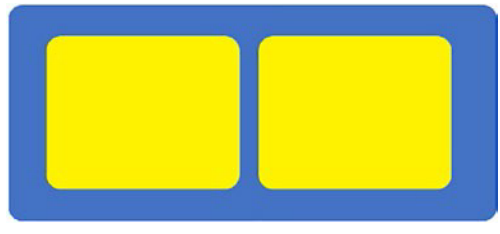


Рис. 4-12. *Контейнер флексбокс, расширяющийся на строку*

Для этого мы используем **flex-direction: rows**:

```
.flex-container { display:
  flex;
```

```
  flex-direction: row;
}
```

Если мы хотим, чтобы элементы располагались друг над другом столбцами (Рис. 4-13),

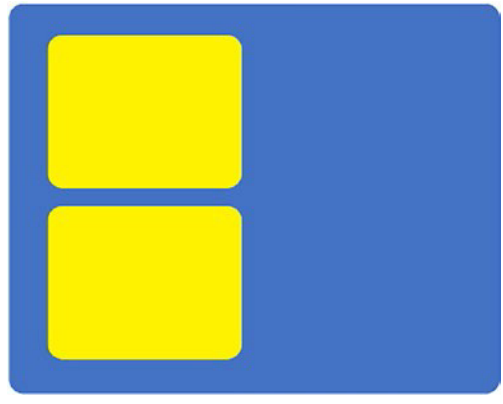


Рис. 4-13. *Контейнер флексбокс, расширяющийся на столбец*

Для этого мы используем **flex-direction: columns**:

```
.flex-container {
  display: flex;
  flex-direction: column;
}
```

Мы можем указать, хотим ли мы переносить элементы или нет. Здесь элементы будут перенесены в следующую строку, если размер экрана меньше исходного (Рис. 4-14).

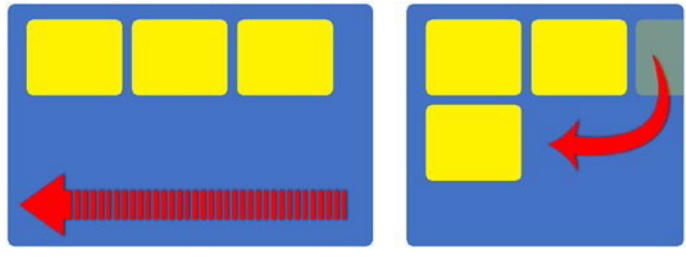


Рис. 4-14. *Перенос контейнера флексбокс*

В этом примере мы собираемся перенести содержимое, поэтому добавляем **flex- wrap: wrap**:

```
.flex-container { display:
  flex;
  flex-direction: row;
  flex-wrap: wrap;
}
```

Теперь, когда мы создали контейнер, мы можем добавить некоторые элементы, используя элемент . В наш HTML-код мы можем добавить

```
<div class="flex-container">
  <div> This is content inside item </div>
  <div> <img src="img/bluebox.png"> </div>
  <div> <img src="img/bluebox.png"> </div>
</div>
```

Свойство order определяет порядок флекс-элементов:

```
<div class="flex-container">
  <div style="order: 3"> This is content inside item </div>
  <div style="order: 1"> <img src="img/bluebox.png"> </div>
  <div style="order: 2"> <img src="img/bluebox.png"> </div>

</div>
```

Если мы хотим стилизовать элементы внутри контейнера, мы можем использовать символ «>»:

```
.flex-container-name > child-item-name {
...
}
```

Этот символ используется для выбора элемента с определенным

родителем. Поскольку каждый элемент во флекс-контейнере указан с помощью элемента , это дочерние элементы. Итак, нам нужно наименование дочернего элемента — div, и оно принадлежит flex-container. Теперь давайте изменим цвет фона на светло-серый, затем добавим поля и отступы, чтобы распределить элементы:

```
.flex-container > div {
  background-color: #f1f1f1;
  margin: 10px;
  padding: 10px;
}
```

На Рис. 4-15 наш HTML-код расположен вверху слева, а наш CSS-код расположен слева внизу. Справа мы можем увидеть, как все это выглядит в веб-браузере. Что произойдет, если вы измените размер окна браузера (Рис. 4-15)?

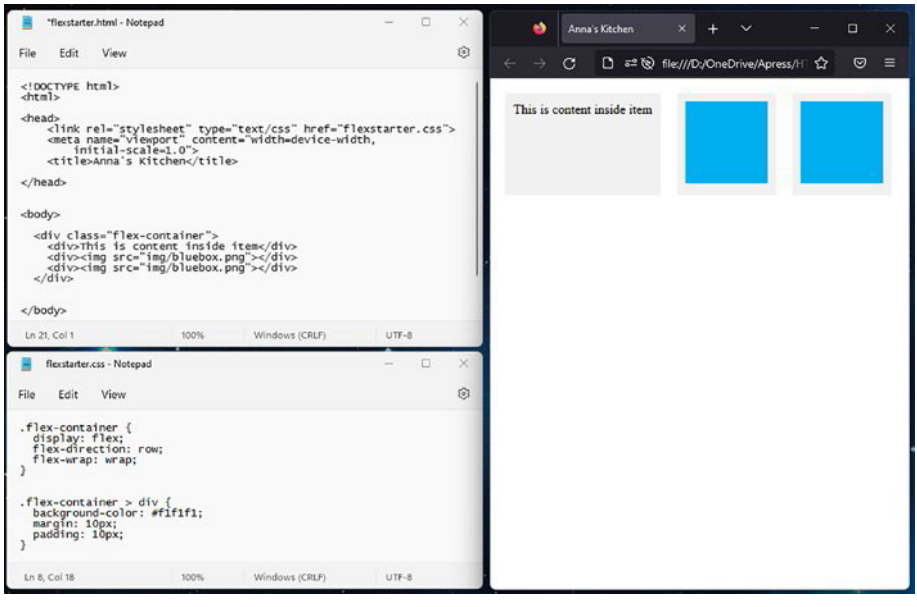


Рис. 4-15. *Изменение размера*

Применение флексбокса на практике

Давайте создадим макет нашего сайта (Рис. 4-16). Здесь мы хотим создать заголовок и панель навигации в верхней части страницы; затем внизу мы хотим расположить контент сайта с боковой панелью справа — это единственная часть, где мы используем флексбокс. Наконец, мы хотим добавить на страницу нижний колонтитул.

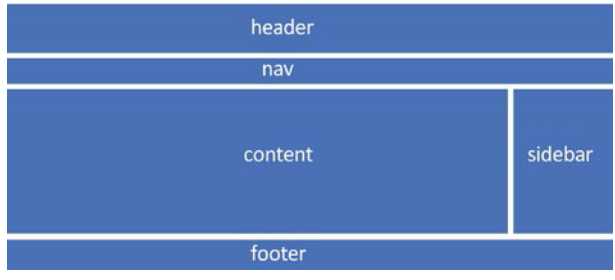


Рис. 4-16. *Макет веб-сайта*

В файле кода CSS мы можем объявить наш заголовок, навигацию и нижний колонтитул, используя простые селекторы классов CSS:

```
.header {
  background: orange;
  padding: 2em;
  text-align: center;
}

.nav {
  background: yellow;
  padding: 1em;
  text-align: center;
}

.footer {
  background: orange;
```

```
  padding: 1em;
}
```

Когда мы добиремся до контента и боковой панели, где мы хотим использовать флексбокс. Мы можем создать наш контейнер:

```
.flex-container {
  display: flex;
  flex-direction: row;
}
```

Затем создадим содержимое и боковую панель как дочерние элементы контейнера. Добавим отступы, чтобы выделить содержимое, и установим серый цвет фона боковой панели:

```
.flex-container > .content {
  padding: 10px;

}

.flex-container > .sidebar {
  background-color: #f1f1f1;
  padding: 10px;

}
```

Теперь в наш файл HTML-кода мы можем добавить заголовок и панель навигации, используя элемент . Просто добавим имя селектора, определяющего заголовок, используя атрибут «class» в открывающем теге . Добавим контент для отображения между открывающим и закрывающим тегами :

```
<div class="header">
      <h1> Welcome To Anna's Kitchen</h1>
      <b> The Home of the Roast! </b>
</div>
```

Сделаем то же самое для панели навигации:

```
<div class="nav">
    <a href="index.html"> Home </a> |
    <a href="menu.html"> Menu </a> |
    <a href="book.html"> Book a Table </a>
</div>
```

Далее нам нужно создать флексбокс-контейнер, в котором будет находиться содержимое и боковая панель. Мы можем добавить флекс-контейнер, используя атрибут «class» в открывающем теге :

```
<div class="flex-container">
```

Затем внутри элемента флекс-контейнера, который мы объявили ранее, мы можем добавить еще один элемент с дочерним элементом содержимого, используя атрибут «class» в открывающем теге :

```
<div class="content">
    <p>
    Every day, our expert chefs prepare a mouth-watering
    feast of hand-carved meats including <b> beef,
    turkey, pork and marmalade-glazed gammon.</b> All
    accompanied by ruffled roasties, a wide range of veg,
    and giant yorkshire puddings. <i> Served up by our
    friendly team, every day of the week, our
    mouthwatering Sunday roast and weekday carvery are
    sure to be a hit with all the family.</i> </p>
</div>
```

Сделаем то же самое с боковой панелью:

```
<div class="sidebar">
    <img src="img/menumap.png" width="220px"
    usemap="#foodmenu">
    <map name = "foodmenu">
      <area shape= "rect" coords = "0,0,220,202" href =
      "menu1.htm">
      <area shape= "rect" coords = "0,202,220,395" href =
      "menu2.htm">
      <area shape= "rect" coords = "0,395,220,596" href =
      "menu3.htm">
    </map>
</div>
```

Не забудьте закрыть флекс-контейнер с помощью **flex-container**

```
</div>
```

Мы увидим что-то вроде Рис. 4-17.

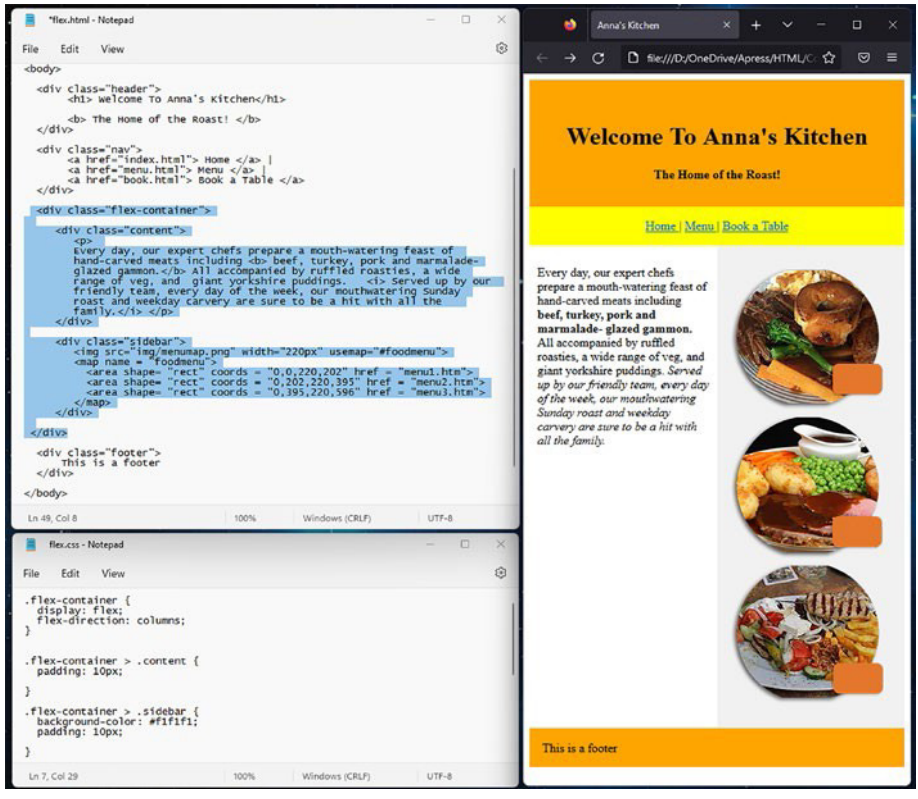


Рис. 4-17. *Флекс-контейнер*

Обратите внимание, что при изменении размера окна ничего не переносится. Чтобы сделать перенос, нам нужно добавить атрибут flex в контейнер:

```
.flex-container {
  display: flex;
  flex-direction: row;
  flex-wrap: wrap;
}
```

а также в дочерние элементы:

```
flex: flex-grow flex-shrink flex-basis;
```

flex-grow определяет, насколько элемент увеличивается по отношению к остальным флекс-элементам. **flex-grow: 0** означает, что элементы не будут увеличиваться. **flex-grow: 1** означает, что элементы могут увеличиться по сравнению с их исходным размером.

flex-shrink определяет, насколько элемент уменьшится относительно остальных флекс-элементов. **flex-shrink: 1** означает, что элементы могут уменьшаться по сравнению с их исходным размером.

flex-basis — это длина элемента, измеряемая в «%», «px» или «em». Вы также можете указать значения: “auto” или “inherit.”

Например, если я добавлю следующее в элемент контента:

```
flex: 1 1 250px;
```

Что это значит?

flex-grow: 1 означает, что элементы могут увеличиться по сравнению с их исходным размером.

flex-shrink: 1 означает, что элементы могут уменьшаться по сравнению с их исходным размером.

flex: 250px означает, что как только первая строка достигает точки, в которой недостаточно места для размещения другого элемента размером 250 пикселей, для элементов создается новая флекс-строка. По мере того, как элементы могут увеличиваться, они увеличиваются более чем на 250 пикселей, чтобы полностью заполнить каждую строку. Если в последней строке только один элемент, он растянется на всю строку.

Итак, мы заканчиваем следующим:

```
.flex-container > .content {
  flex: 1 1 250px;
  padding: 10px;
}

.flex-container > .sidebar {
```

```
  flex: 1 1 50px;
  background-color:
  #f1f1f1; padding: 10px;
}
```

Давайте посмотрим, как это выглядит в браузере (Рис. 4-18).

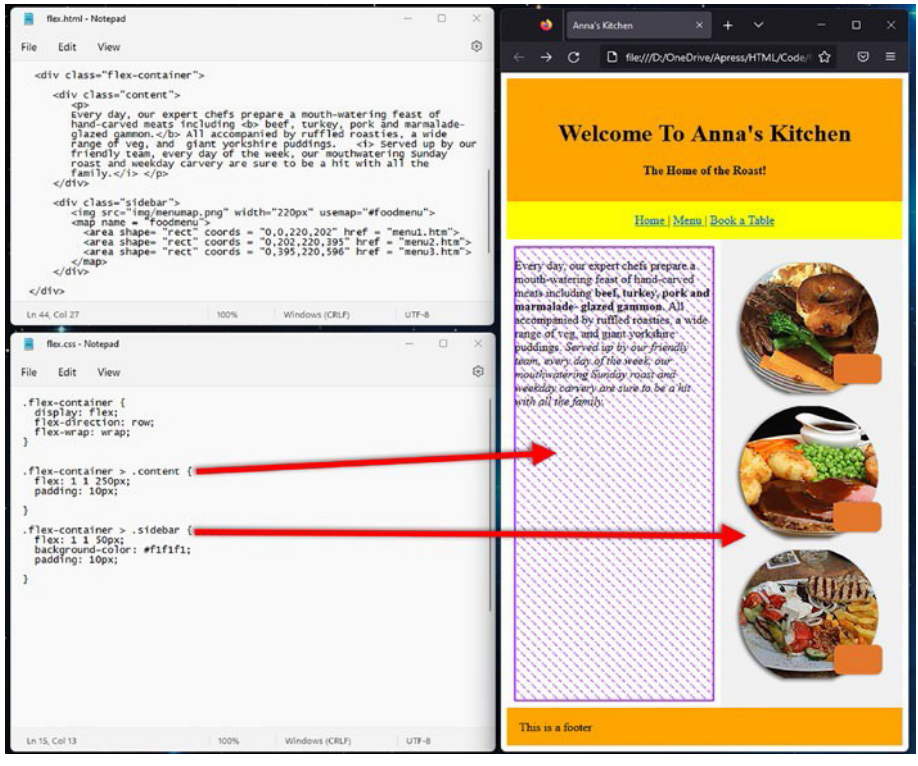


Рис. 4-18. *Флекс-контейнеры в браузере*

Что происходит, когда вы меняете размер окна браузера? Обратите внимание, как они растягиваются, когда мы увеличиваем браузер в ширину. Также обратите внимание, как переворачивается боковая панель, когда мы уменьшаем браузер в ширину. Она будет перенесен, когда содержимое достигнет 250 пикселей (Рис. 4-19).

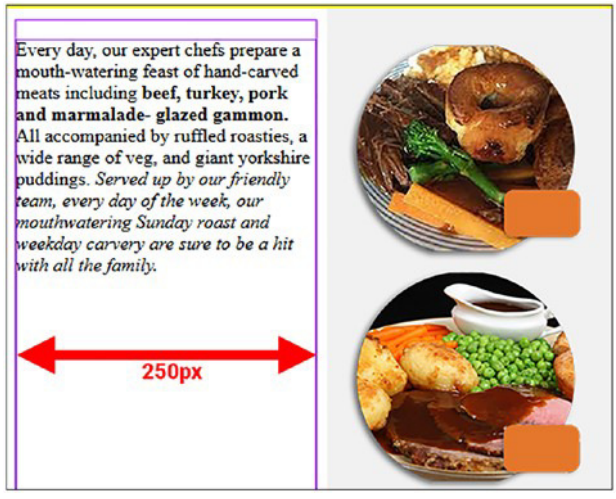


Рис. 4-19. *Боковая панель размером 250 пикселей*

Давай попробуем. Мы уменьшили и увеличили браузер в ширину, чтобы посмотреть, что произойдет (Рис. 4-20).

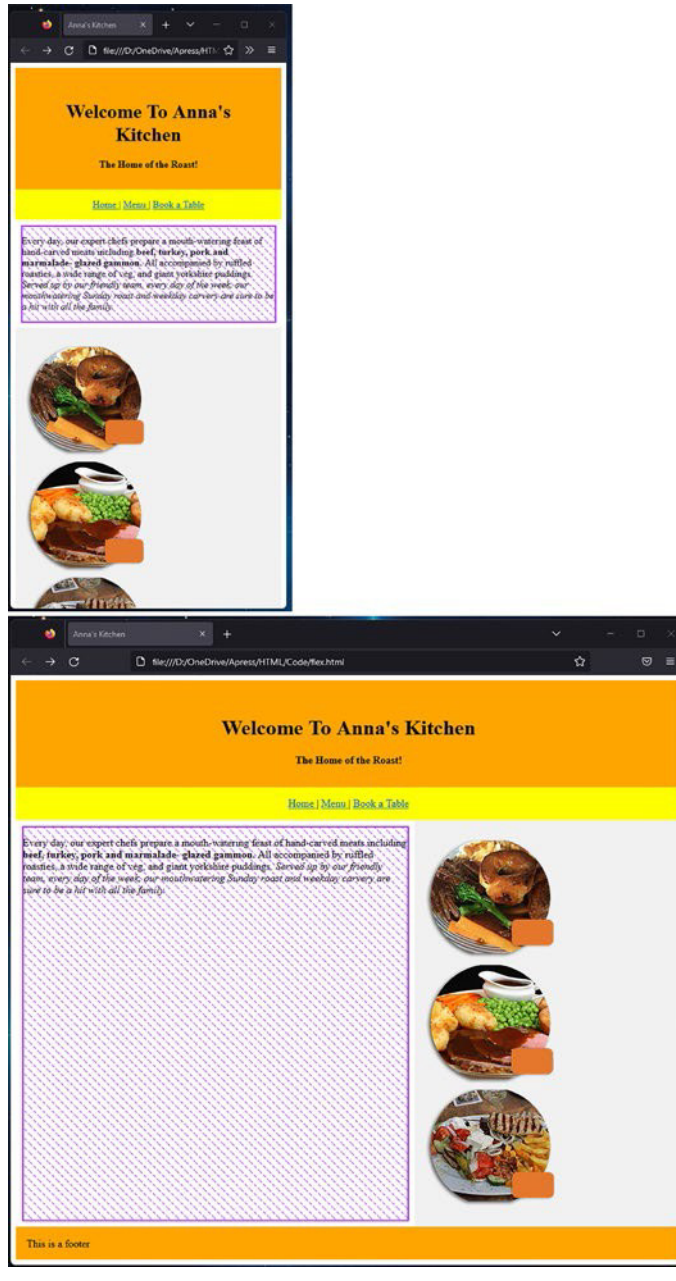


Рис. 4-20. *Уменьшение и увеличение ширины браузра*

Имейте в виду, что флексбокс может обрабатывать только строки или только столбцы, но не то и другое одновременно.

CSS-сетка

Макет CSS-сетки — это двухмерная система макетов на основе сетки со строками и столбцами, которая предназначена для упрощения компоновки веб-страниц, что позволяет разработчику выравнивать элементы по столбцам и строкам. CSS-сетка — идеальный кандидат для макета всей страницы.

На Рис. 4-21 вертикальные линии называются столбцами, а горизонтальные линии — строками. Пробелы между каждым столбцом/строкой называются пробелами. Элементы сетки, такие как изображения или текст, могут быть выровнены по строкам и столбцам сетки и называются элементами сетки (Рис. 4-22).

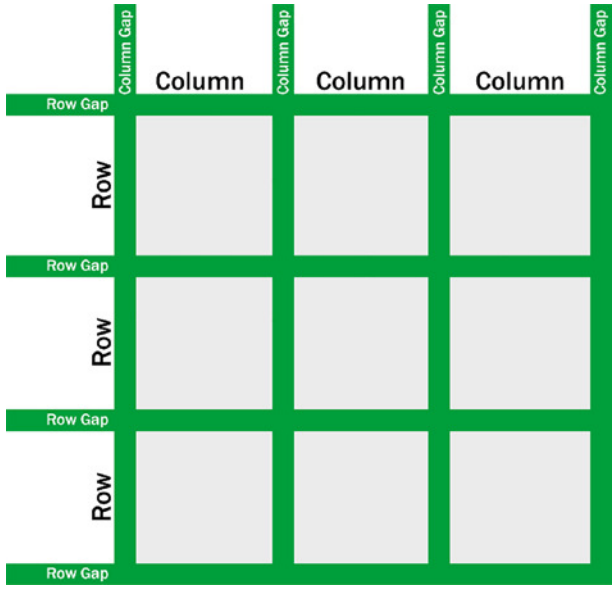


Рис. 4-21. *CSS-сетка*

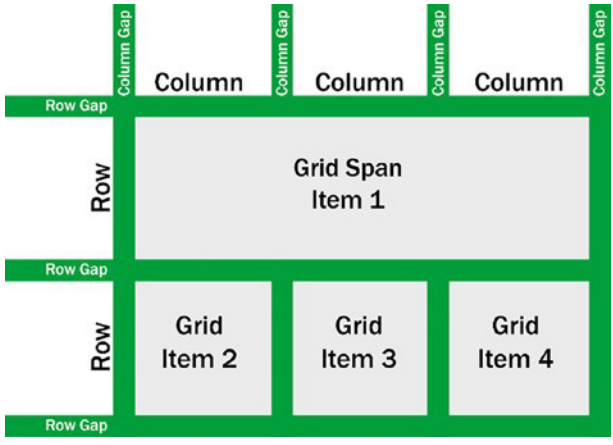


Рис. 4-22. *CSS-сетка с элементами*

Применение CSS-сетки на практике

В этом примере мы хотим сконфигурировать наш веб-сайт следующим образом. Мы хотим, чтобы ширина составляла четыре столбца. Это означает, что верхний и нижний колонтитулы будут занимать все четыре столбца. Мы также хотим, чтобы контент заполнял три столбца (столбец 1, столбец 2, столбец 3). Боковая панель займет последний столбец (столбец 4). Сайт также будет разделен на три ряда. Заголовок в ряду 1, содержимое и боковая панель во втором ряду и нижний колонтитул в ряду 3. Мы можем видеть этот макет на Рис. 4-23.

Рис. 4-23. *Шаблон с CSS-сеткой*

Во-первых, нам нужно создать контейнер сетки:

```
.container  {
   display:grid
}
```

Далее нам нужно указать, сколько строк и столбцов мы собираемся использовать. Исходя из шаблона на Рис. 4-23, нам нужно четыре столбца. Свойство Grid-template-columns определяет количество столбцов в макете сетки. Атрибут «auto» означает, что размер столбцов будет изменен автоматически. Нам необходимо

указать размер для каждого столбца. Здесь также можно указать размер:

```
.container { display:grid
  grid-template-columns: auto auto auto auto;
}
```

Теперь нам нужно указать, сколько рядов мы хотим использовать. Судя по Рис. 4-23, нам нужно три ряда: один для заголовка, один для содержимого и боковой панели и еще один для нижнего колонтитула. Свойство Grid-template-rows определяет количество рядов в макете сетки. Атрибут «auto» означает, что размер рядов будет автоматически изменен. Вам необходимо указать размер для каждого ряда. Здесь также можно указать размер:

```
.container { display:grid
  grid-template-columns: auto auto auto auto;
  grid-template-rows: auto auto auto;
}
```

Теперь, когда мы создали наш контейнер, нам нужно создать элементы. Первый элемент — заголовок. Здесь мы указали, что хотим, чтобы фон заголовка был оранжевым. Далее мы хотим указать, в каком ряду/столбце начинается заголовок:

```
grid-row-start: 1;
grid-column-start: 1;
```

Здесь мы начинаем с ряда 1, столбца 1, а заголовок заканчивается перед рядом 2, столбцом 5. Для этого мы используем свойство Grid-row-end. Оно определяет, сколько рядов будет охватывать элемент или на какой строке ряда он закончится:

```
grid-row-end: 2;
grid-column-end: 5;
```

Мы заканчиваем так:

```
.header {
    background-color: orange;
    grid-row-start: 1;
    grid-column-start: 1;
    grid-row-end: 2;
    grid-column-end: 5;
    padding: 20px;
    text-align: center;
}
```

Наконец, мы можем добавить отступы, чтобы раздвинуть содержимое заголовка и выровнять текст по центру.

Мы можем сделать то же самое и для нижнего колонтитула. Здесь нижний колонтитул начинается в ряду 3 и заканчивается после столбца 4 (то есть 5). Опять же, мы меняем цвет фона на оранжевый и добавляем отступы, чтобы текст был выровнен по центру:

```
.footer {
    background-color: orange;
    grid-row-start: 3;
    grid-column-start: 1;
    grid-row-end: 4;
    grid-column-end: 5;
    padding: 20px;
    text-align: center;
}
```

Теперь, что касается содержимого, оно начнется с ряда 2, но охватит только столбцы 1, 2 и 3, поэтому мы закончим после 3 (то есть 4). Боковая панель заполнит последний столбец:

```
.content {
    grid-row-start: 2;
```

```
    grid-column-start: 1;
    grid-row-end: 3;
    grid-column-end: 4;
    padding: 20px;
}
```

Что касается боковой панели, мы хотим начать с ряда 2, просто заполнить последний столбец после содержимого, поэтому мы начинаем со столбца 4 и заканчиваем после столбца 4 (то есть 5):

```
.sidebar {
    background-color: lightgrey; grid-row-
    start: 2;
    grid-column-start: 4;
    grid-row-end: 3;
    grid-column-end: 5;
}
```

Теперь нам нужно создать страницу, используя контейнеры сетки нашего HTML-документа. Мы сделаем это с помощью тега .

Сначала мы добавляем контейнер; помните, что мы назвали контейнер layout-grid и он объявлен как класс, поэтому мы добавляем его с помощью атрибута class к открывающему тегу :

```
<div class="layout-grid">
```

Внутри мы можем добавить элементы. Сначала идет заголовок:

```
<div class="header">
  <h1> Welcome To Anna's Kitchen</h1>
  <b> The Home of the Roast! </b>
</div>
```

затем идут остальные элементы. Просто добавьте содержимое к элементам между двумя тегами :

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

```
        ...
    </div>

    <div class="sidebar">
        ...
    </div>

    <div class="footer">
        ...
    </div>
</div>
```

Давайте соберем все это вместе и посмотрим, как это выглядит (Рис. 4-24).

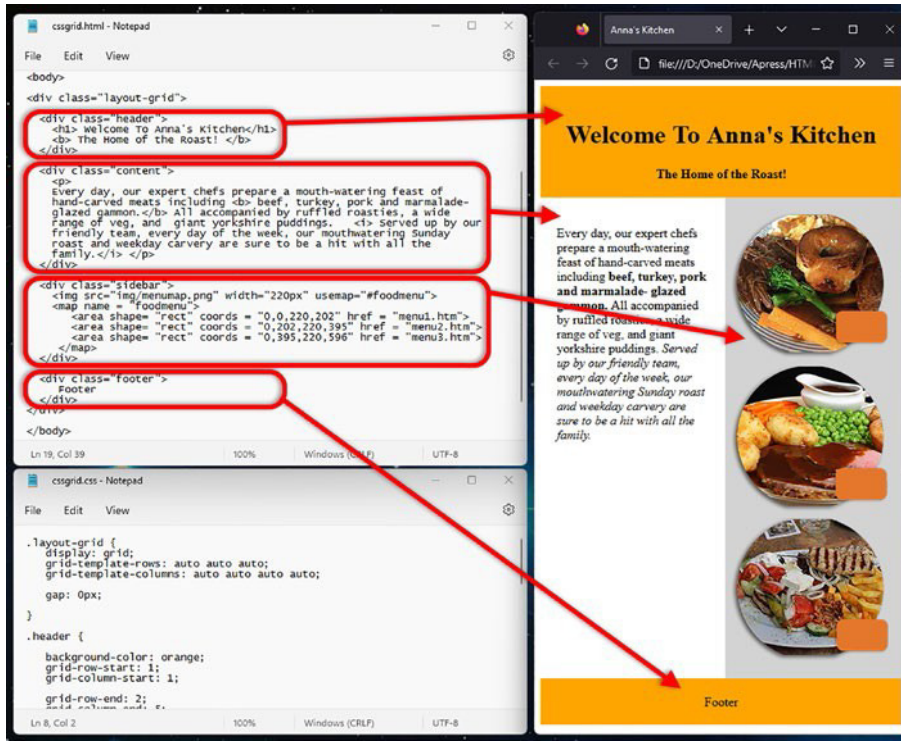


Рис. 4-24. *Добавленые элементы*

Адаптивные макеты сетки

Адаптивный дизайн — это подход к веб-дизайну, который позволяет контенту вашего веб-сайта адаптироваться к различным размерам экрана и окон, используемым на различных устройствах, таких как телефоны, планшеты и компьютеры.

Если мы откроем простой веб-сайт на планшете, экран выглядит нормально и имеет правильный размер (Рис. 4-25).

Рис. 4-25. *Веб-страница на планшете*

Обратите внимание, что происходит, когда вы меняете размер окна браузера на ПК. Макет начнет растягиваться и изменять размеры. Вы можете видеть, что фоновое изображение на боковой панели справа больше не помещается и стало слишком маленьким (Рис. 4-26).

Рис. 4-26. *Веб-сайт на ноутбуке*

Или, если мы просматриваем сайт на телефоне (Рис. 4-27), боковая панель может быть скрыта под основным текстом.

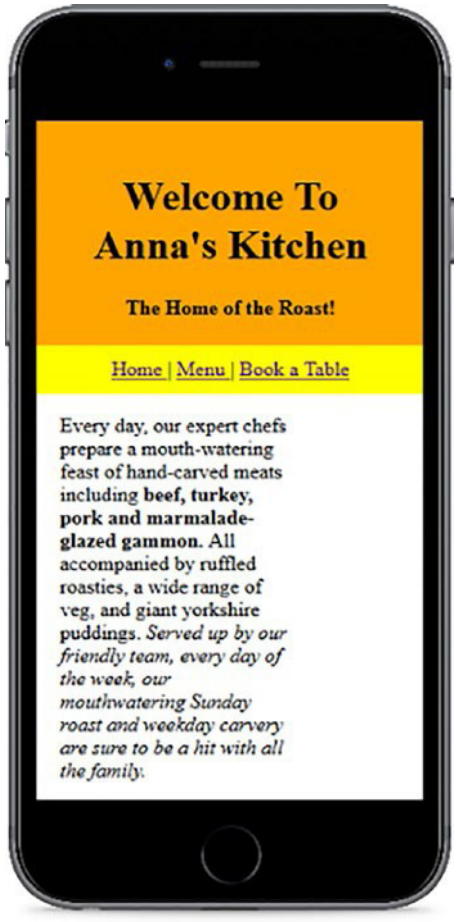


Рис. 4-27. *Веб-сайт на смартфоне*

Макет меняется в зависимости от размера экрана. Страницы должны быть оптимизированы для экранов разных размеров, но как это сделать? В наши дни, благодаря смартфонам, планшетам, ноутбукам с экранами разных размеров и ПК, многие веб-страницы построены на адаптивном виде сетки. Это означает, что веб-страницы, основанные на сетке, разделены на столбцы.

Адаптивное представление сетки часто имеет 12 столбцов и общую ширину 100%. Она будет уменьшаться или увеличиваться при изменении размера окна браузера (Рис. 4-28).

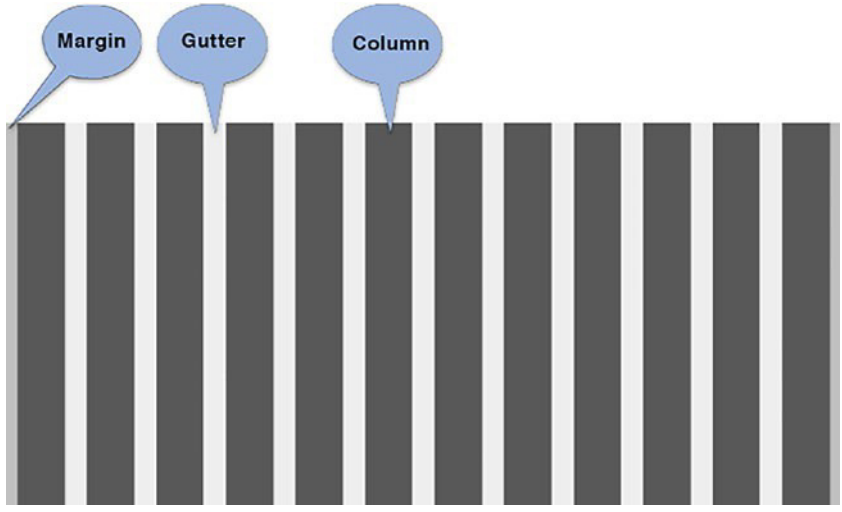


Рис. 4-28. *Адаптивный вид сетки*

Между каждым столбцом вы обнаружите промежуток, а по обе стороны сетки присутствуют поля.

Чтобы создать представление сетки в CSS, мы сначала вычисляем процент для одного столбца:

100%/12**x1**столбцов=8,33%.

Следующий столбец будет100%/12**x2**=16,66%.

Следующий столбец: 100%/12**x3**=25%. И так далее….

Далее нам нужно создать по одному классу для каждого из 12 столбцов и размеров в процентах, которые мы рассчитали на предыдущем шаге:

```
.col-1 {width: 8.33%;}
.col-2 {width: 16.66%;}
.col-3 {width: 25%;}
.col-4 {width: 33.33%;}
.col-5 {width: 41.66%;}
.col-6 {width: 50%;}
.col-7 {width: 58.33%;}
```

```
.col-8 {width: 66.66%;}
.col-9 {width: 75%;}
.col-10 {width: 83.33%;}
.col-11 {width: 91.66%;}
.col-12 {width: 100%;}
```

Это создаст сетку, подобную приведенной на Рис. 4-29.

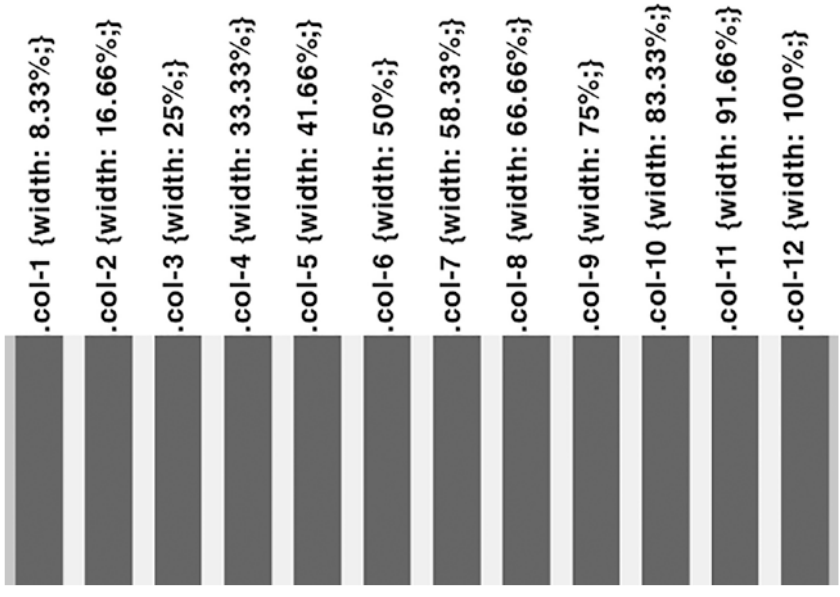


Рис. 4-29. *Адаптивный вид сетки, показывающий ширину столбцов*

Помните файл CSS, который мы создали в предыдущем разделе. Отступы и границы должны включаться в общую ширину и высоту элементов. Чтобы обеспечить это, мы добавляем следующее в начало файла CSS:

```
* {
  box-sizing: border-box;
}
```

Мы можем добавить объявления столбцов в конец файла CSS:

```
.col-1 {width: 8.33%;}
.col-2 {width: 16.66%;}
.col-3 {width: 25%;}
.col-4 {width: 33.33%;}
.col-5 {width: 41.66%;}
.col-6 {width: 50%;}
.col-7 {width: 58.33%;}
.col-8 {width: 66.66%;}
.col-9 {width: 75%;}
.col-10 {width: 83.33%;}
.col-11 {width: 91.66%;}
.col-12 {width: 100%;}
```

Теперь мы можем создать страницу с помощью HTML. Мы можем добавить заголовок и панель навигации. Эти два раздела будут занимать 100 % всей страницы:

```
<header>
    <h1> Welcome To Anna's Kitchen</h1>

    <b> The Home of the Roast! </b>
</header>

<nav>
   <a href="index.html"> Home </a> |
   <a href="menu.html"> Menu </a> |
   <a href="menu.html"> Book a Table </a>
</nav>
```

Далее давайте добавим контент и боковую панель. Теперь для этого раздела мы хотим, чтобы раздел контента занимал восемь столбцов, а боковая панель - оставшиеся четыре столбца:

```
<content class="col-8">
```

```
    <p>
    Every day, our expert chefs prepare a mouth-watering
    … </p>
    <img src="img/carvery.jpg" width="365px">
</content>
```

Вы можете видеть, что основной контент будет занимать восемь столбцов, как это показано на Рис. 4-30.

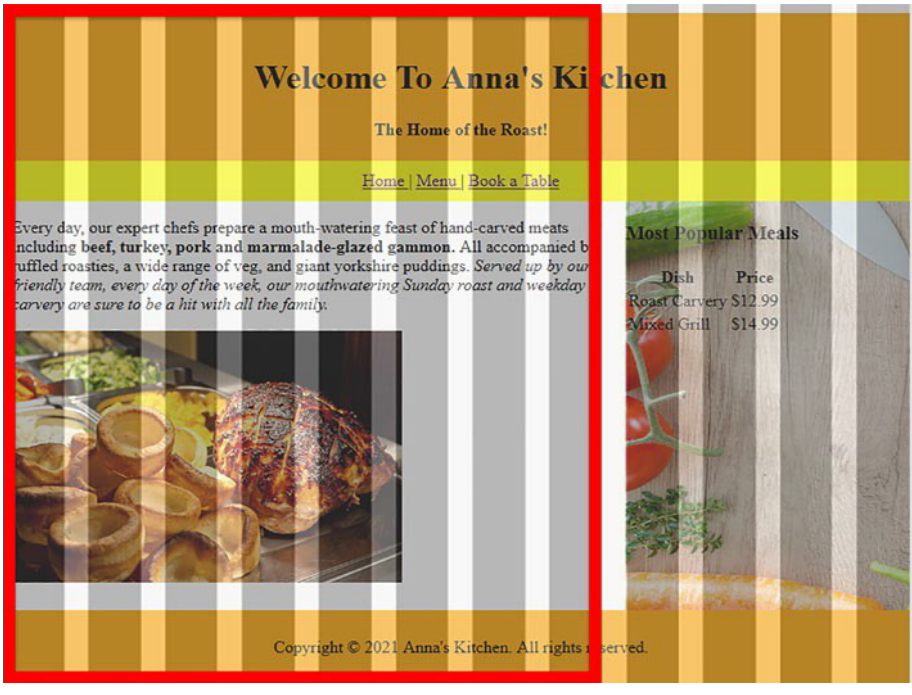


Рис. 4-30. *Адаптивный вид сетки для основного контента*

После этого мы можем добавить боковую панель:

```
<sidebar class="col-4">
    <h3 align="center">Most Popular Meals</h3>
    <table align="center">
      <tr>
        <th>Dish</th>
```

```
        <th>Price</th>
      </tr>
      <tr>
        <td>Roast Carvery</td>
        <td>$12.99</td>
      </tr>
      <tr>
        <td>Mixed Grill</td>
        <td>$14.99</td>
    </tr>
</table>
</sidebar>
```

Она охватывает четыре столбца, как показано на рисунке 4-31.

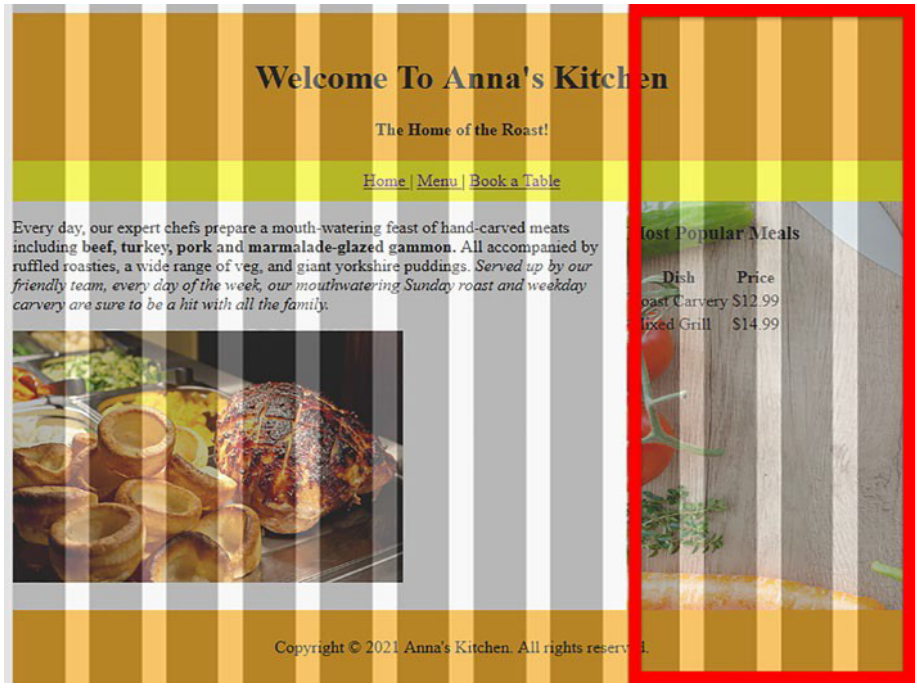


Рис. 4-31. *Адаптивный вид сетки для боковой панели*

Что же произойдет, если вы измените размер окна браузера? Вы увидите, как содержимое растянется на весь экран (Рис. 4-32).

Рис. 4-32. *Содержимое растянуто на весь экран*

Что происходит, когда мы просматриваем сайт на телефоне? Представление на телефоне, как показано на Рис. 4-33, немного сжатое, и его можно было бы использовать с некоторыми точками разрыва, чтобы переместить боковую панель вниз.

Рис. 4-33. *Сжатый просмотр на телефоне*

Для этого мы должны понять, что такое область просмотра. Область просмотра — это часть веб-сайта, которую может видеть пользователь (Рис. 4-34).

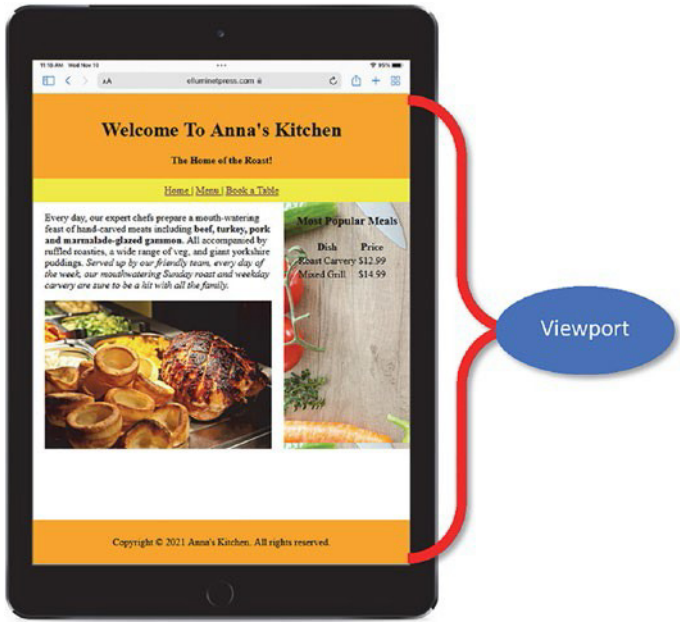


Рис. 4-34. *Область просмотра*

Мета-тег области просмотра дает браузеру инструкции по управлению размерами и масштабированием страницы (Рис. 4-35):

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

Используя значение мета области просмотра, **width=device-width** указывает странице соответствовать ширине экрана в пикселях, независимых от устройства. Часть **initial-scale=1.0** устанавливает начальный уровень масштабирования при первой загрузке страницы браузером.

Чтобы обеспечить наилучшее взаимодействие, мобильные браузеры отображают страницу на экране настольного компьютера шириной около 767–980 пикселей, а затем масштабируют содержимое, увеличивая размер шрифта и изменяя размер содержимого в соответствии с размером экрана.

Рис. 4-35. *Веб-сайт на экранах разных размеров*

Медиа-запросы позволяют отвечать клиентскому браузеру с настраиваемым отображением для определенных размеров области просмотра. Правило @media включает блок свойств CSS только в том случае, если определенное условие истинно.

Возвращаясь к нашему CSS, мы можем задать ширину всех столбцов так, чтобы они охватывали 100% ширины на смартфонах с маленькими экранами. Следующий код выбирает любой элемент, который содержит **“col-”** в любом месте значения атрибута класса:

```
[class*="col-"]
  { width:
  100%;
}
```

Это называется селектором атрибутов CSS, в котором атрибут, который мы хотим выбрать, заключен в квадратные скобки [].

Вот медиа-запрос для работы с большими экранами. Свойство мультимедиа min-width определяет минимальную ширину конкретного устройства. Таким образом, ширина экрана должна быть 768 пикселей или больше. Это будет хорошо работать на больших планшетах, таких как iPad или ноутбук.

```
@media only screen and (min-width: 768px) {
  .col-1 {width: 8.33%;}
  .col-2 {width: 16.66%;}
  .col-3 {width: 25%;}
  .col-4 {width: 33.33%;}
  .col-5 {width: 41.66%;}
  .col-6 {width: 50%;}
  .col-7 {width: 58.33%;}
  .col-8 {width: 66.66%;}
  .col-9 {width: 75%;}
  .col-10 {width: 83.33%;}
  .col-11 {width: 91.66%;}
  .col-12 {width: 100%;}
}
```

Это создает точку разрыва, когда ширина окна браузера составляет 768 пикселей.

Вы можете установить точки разрыва, используя свойства min-width и max-width. Когда следует использовать каждый из них? Что ж, если вы сначала разрабатываете макет для небольших экранов смартфонов, тогда используйте точки разрыва минимальной ширины и постепенно увеличивайте его. Если вы сначала разработали веб-сайт для отображения на настольном компьютере и хотите адаптировать макет для экранов меньшего размера, используйте максимальную ширину и переходите к самому маленькому экрану.

Упражнение

1. Что такое CSS?
2. Как включить CSS в свой HTML-документ? Опишите три метода.

3. Что такое селектор? Какие бывают типы и что они делают?
4. Создайте новый HTML-файл и назовите его ch04.html.
5. Создайте новый файл CSS и назовите его ch04.css.
6. В файл ch04.html добавьте базовую структуру HTML-документа.
7. Свяжите CSS-файл ch04.css с вашим HTML-документом.
8. В чем разница между абсолютными и относительными измерениями?
9. Назовите некоторые абсолютные единицы измерения.
10. Назовите некоторые относительные единицы измерения.

Заключение

- Каскадные таблицы стилей (CSS) используются для определения и настройки стилей и макетов ваших веб-страниц.
- Может быть встроено в элемент HTML.
- Можно включить из внешнего файла с помощью <script href= “”>.
- Может быть включен в сам HTML-файл между тегами <script>… </script>.
- Селектор типа создает общий стиль для объявленного элемента.
- Селектор классов используется для применения стилей к определенному элементу HTML и на них ссылаются с помощью точки.

- ID-селектор нацелен на один элемент, может использоваться только один раз на странице и ссылается на него с помощью хэша.
- Универсальный селектор соответствует каждому элементу на странице и обозначается звездочкой.
- Укажите цвета, используя Ключевое слово, Шестнадцатеричное значение или значение RGB.
- Используйте дюймы, миллиметры, пиксели или точки для указания абсолютных размеров.
- Используйте em или проценты для указания относительных размеров.
- Флксбокс — модуль макета, предназначенный для размещения групп элементов в одном направлении (с использованием ряда или столбца, но не того и другого одновременно).
- CSS-сетка – это функция двумерного макета, которая идеально подходит для макетов всей страницы (с использованием рядов и столбцов).

ГЛАВА 5

Спецэффекты

Используя HTML и CSS, вы можете добавлять различные эффекты для украшения своего сайта. К ним относятся

- Эффекты при наведении курсора
- Кнопки
- Закругленные углы
- Тени
- Градиенты

Эти эффекты следует использовать с осторожностью, поскольку при их чрезмерном использовании они могут раздражать и отвлекать внимание.

Однако эффекты могут быть полезны для акцентирования внимания на разделе или объекте.

Текстовые эффекты

В этом примере мы собираемся добавить эффект тени к заголовку. Используя CSS, мы можем стилизовать заголовок, используя свойство text-shadow:

```
h1 {
  text-shadow: 2px 2px 2px lightgrey;
}
```

Мы также можем изменить текст на белый, используя свойство цвета:

```
h1 {
  color: white;
  text-shadow: 2px 2px 2px lightgrey;
}
```

Давайте рассмотрим пример. Здесь мы добавили свойство text-shadow к селектору заголовка 1, как вы можете видеть в файле about.css (Рис. 5-1).

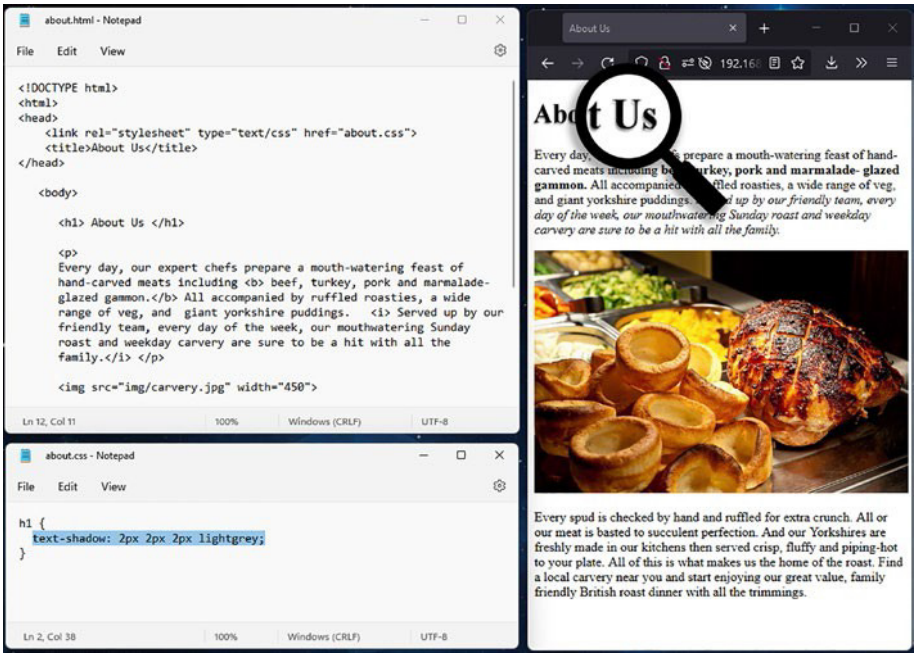


Рис. 5-1. *Свойство «Тень от текста»*

В веб-браузере справа на Рис. 5-1 вы можете увидеть тень вокруг заголовка.

Закругленные углы изображения

Мы можем стилизовать изображения. Мы можем закруглить углы изображения. Величина изгиба угла называется радиусом границы:

```
img {
  border-radius: 10px;
}
```

Давайте рассмотрим пример. Здесь мы добавили свойство border-radius в селектор img, как вы можете видеть это в файле about.css на Рис. 5-2.

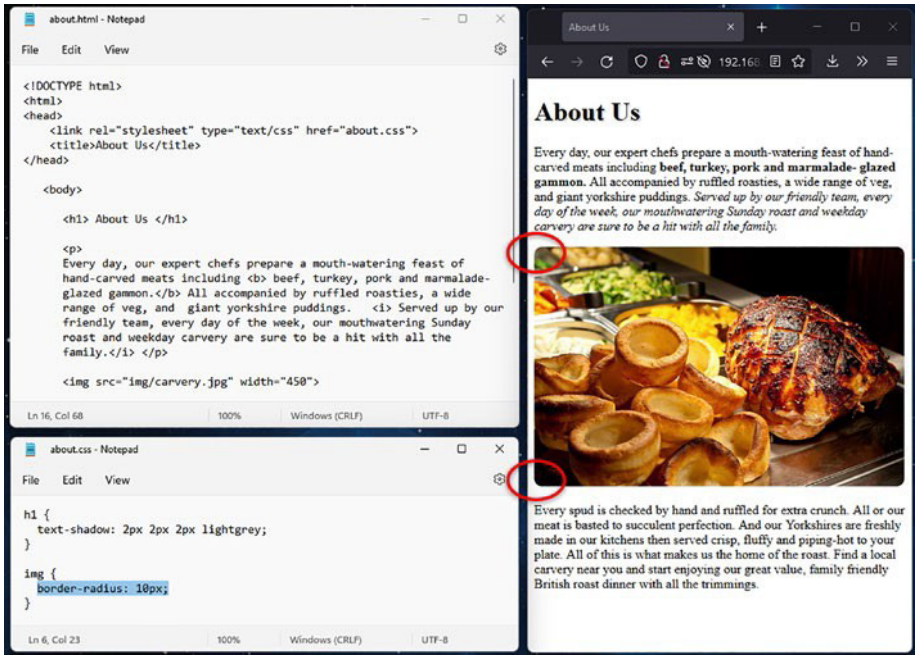


Рис. 5-2. *Закругленные углы*

Вы можете увидеть на рис. 5-2 закругленные углы фотографии в окне браузера.

Кнопки

Добавляем стили к кнопкам. Здесь мы создали селектор класса кнопок. Мы установили для кнопки оранжевый цвет фона, белый цвет для текста, а также добавили закругленные углы с помощью свойства border-radius. Мы также добавили свойство для изменения указателя мыши на указатель руки с помощью свойства курсора:

```
.button {
  background-color: orange;
  color: white;
  border: none; border-
  radius: 4px; padding:
  15px 25px; cursor:
  pointer;
}
```

Далее мы добавили для кнопки состояние при наведении на нее курсора. Здесь мы установили зеленый цвет фона и белый цвет для текста всякий раз, когда указатель мыши наводится на кнопку:

```
.button:hover {
  background-color: green;
  color: white;
}
```

Наконец, мы добавляем класс для кнопки в HTML-документ:

```
<button class="button">Submit</button>
```

Давайте разберемся. В нижней части страницы в веб-браузере вы увидите, как кнопка меняет цвет, а курсор меняется на указатель в виде руки, когда вы наводите указатель мыши на кнопку (Рис. 5-3).

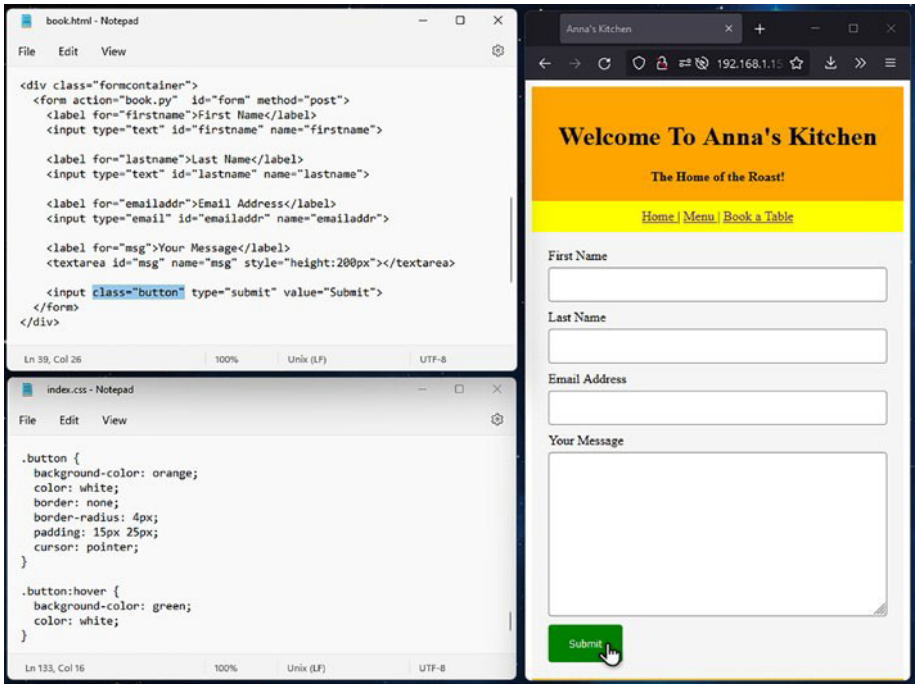


Рис. 5-3. *Курсор меняется*

Градиенты

Вы можете добавить градиенты на свою страницу (рис. 5-4). Градиент — это плавный переход между двумя или более указанными цветами.

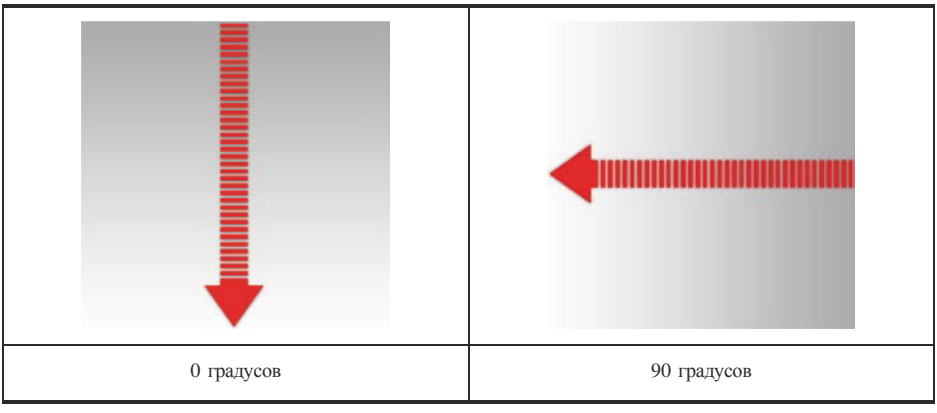


Рис. 5-4. *Градиенты*

Здесь мы определяем линейный градиент, который переходит от белого к светло-серому вниз по странице:

```
#gradient1 {
  background-image: linear-gradient(0deg, white, lightgrey);
}
```

Теперь все, что мы хотим, чтобы охватывал градиент, мы заключаем в теги :

```
<div id="gradient1" style="text-align:center;">

---

</div>
```

Давайте посмотрим на пример. На Рис. 5-5 мы определили ID-селектор градиента. Мы добавили градиент от светло-серого до белого вниз по странице (0 градусов).

Мы также заключили текст между тегами , чтобы знать, где градиент начнется и где заканчивается.

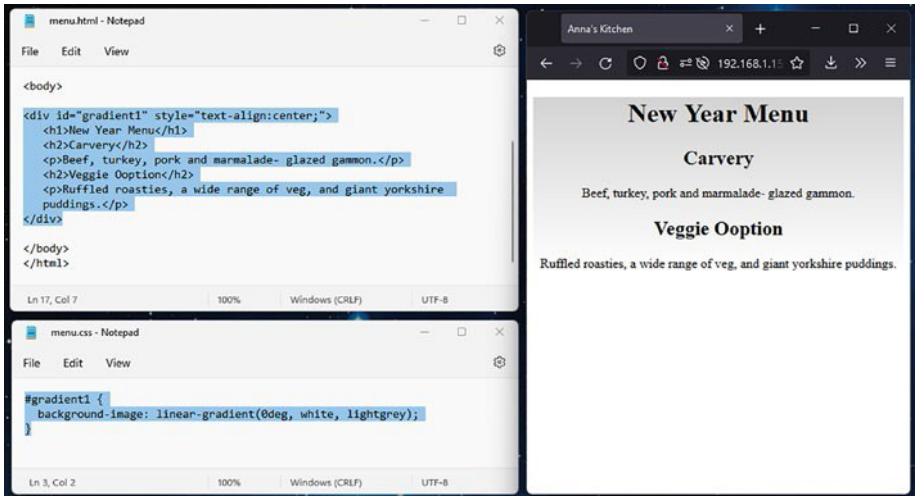


Рис. 5-5. *Градиенты в использовании*

Упражнение

1. Что такое CSS?
2. Что такое селектор? Какие бывают типы и что они делают?
3. Создайте новый HTML-файл и назовите его ch05.html.
4. Создайте новый файл CSS и назовите его ch05.css.
5. В файл ch05.html добавьте базовую структуру HTML-документа.
6. Свяжите CSS-файл ch04.css с вашим HTML-документом.

Заключение

- К элементам HTML можно добавлять различные эффекты.
- Используйте text-shadow, чтобы добавить эффект тени к тексту.
- Используйте border-radius, чтобы скруглить углы изображений и кнопок.
- Используйте кнопку: используйте hover для создания эффекта при наведении курсора.
- Используйте linear-gradient() для создания градиента от одного цвета к другому.

ГЛАВА 6

Мультимедиа

Используя спецификацию HTML5, гораздо проще встроить мультимедиа на свой веб-сайт. Вы можете легко вставлять видео, музыку, анимацию и звук.

Мультимедийные файлы имеют различные расширения.

Аудиофайлы могут иметь формат .wav, .mp3, .aac или .wma.

Видеофайлы могут иметь формат .mp4, .mpg, .wmv, .webm или .avi.

Фотографии и иллюстрации обычно имеют формат .jpg, .png или .webp.

Добавление видео

Используйте внедренные теги <video>…</video>:

```
<video width="450" controls autoplay>
    <source src="video.mp4" type="video/mp4">
</video>
```

Используйте атрибут источника, чтобы указать видеофайл и формат. Здесь вы можете перечислить несколько форматов, но H264 «MP4» и WEBM кажутся наиболее популярными.

Используйте атрибут ширины, чтобы установить ширину видеоокна, или используйте атрибут высоты, чтобы установить его высоту. Обратите внимание: если вы хотите сохранить соотношение сторон видео, вам нужно указать только один из двух атрибутов: ширину или высоту. Это предотвращает растягивание или сжатие

видео. Вы также можете указать процент, например 100 %, чтобы видео занимало всю страницу независимо от размера окна браузера — это иногда полезно, если вы разрабатываете для разных размеров экрана.

Если вы хотите, чтобы такие элементы управления, как воспроизведение, стоп и пропуск, отображались в нижней части окна видео, добавьте атрибут «controls»; если нет, не добавляйте его.

Если вы хотите, чтобы видео автоматически запускалось при загрузке страницы, добавьте атрибут «autoplay». Браузеры имеют тенденцию блокировать эту функцию, и большинство видео не будут воспроизводиться до тех пор, пока пользователь не нажмет кнопку воспроизведения.

Попробуйте добавить код в файл index.html (Рис. 6-1) и посмотрите, что произойдет.

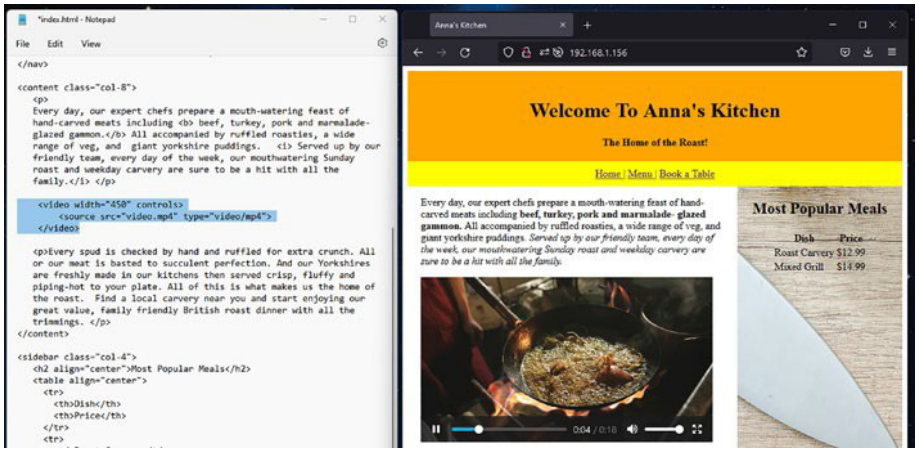


Рис. 6-1. *Внедрение видео*

Если вы загружаете видеофайлы на свой веб-сервер, вам потребуется достаточно места для хранения файлов, а также достаточная пропускная способность для передачи видео всем, кто посещает ваш сайт.

Другой способ — загрузить видеофайл на видеохостинг, например YouTube или Vimeo. Таким образом, ваше видео

передается на устройство пользователя, а не загружается с вашего сайта.

На Рис. 6-2 я загрузил свое видео на Vimeo.

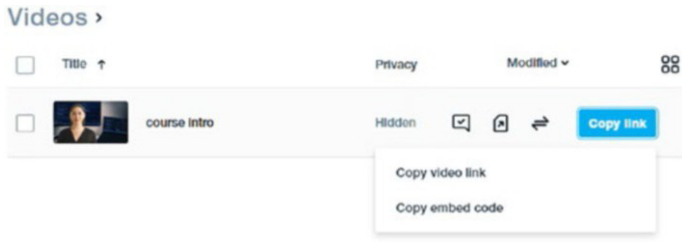


Рис. 6-2. *Загрузка на Vimeo*

Все, что вам нужно сделать, это скопировать код для внедрения видео и вставить его в свой HTML-код:

```
<iframe
    src="https://player.vimeo.com/video/738226502?h=1d66dead64"
    width="450"
    height="220"
    frameborder="0"
    allow="autoplay;
    fullscreen"  allowfullscreen>
</iframe>
```

Если вы используете YouTube, после загрузки видео вам нужно будет скопировать код для встраивания. Для этого выберите «Поделиться» внизу видео на YouTube (Рис. 6-3).

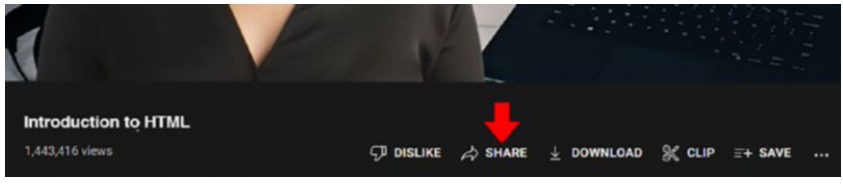


Рис. 6-3. *Кнопка «Поделиться» на YouTube*

Во всплывающем диалоговом окне выберите «Embed» (Рис. 6-4).

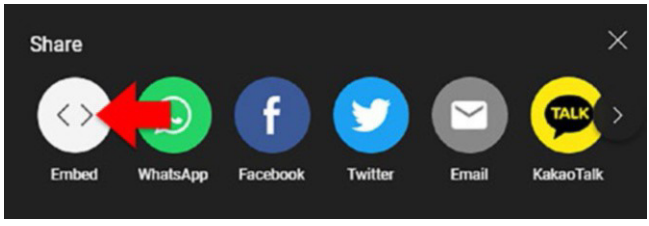


Рис. 6-4. *Embed*

Нажмите «COPY» в правом нижнем углу экрана (Рис. 6-5).

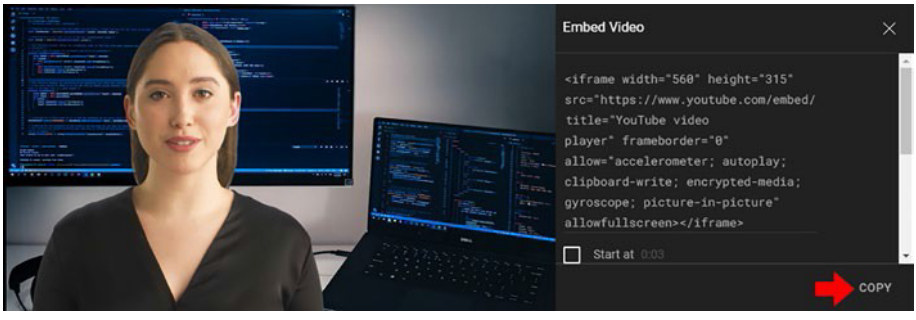


Рис. 6-5. *Копирование кода*

Вставьте код в HTML-документ в то место, где вы хотите разместить видео. Я собираюсь вставить видео после первого абзаца (Рис. 6-6).

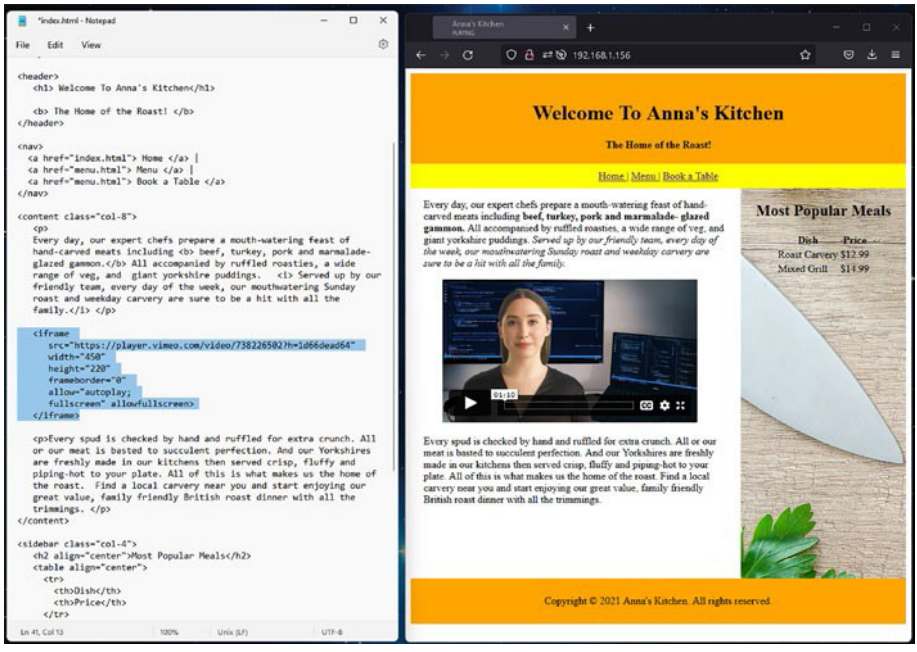


Рис. 6-6. *Видео на сайте*

Вы можете настроить размер видео, используя атрибуты ширины и высоты. Вы также можете добавить или удалить рамку и разрешить автовоспроизведение и полноэкранный режим.

Внедрение аудио

Если вы внедряете фоновую музыку или любой звук на свой веб-сайт, убедитесь, что это уместно и дополняет ваш веб-сайт. Нет ничего хуже, чем зайти на сайт и услышать раздражающую музыку или звук.

Чтобы добавить аудио, используйте теги <audio>…</audio>:

```
<audio controls>
    <source src= "music.mp3" type= "audio/mpeg">
</audio>
```

Атрибут **loop** зацикливает музыку для повторного воспроизведения.

Атрибут **controls** показывает элементы управления воспроизведением, остановкой, треком и громкостью.

Атрибут **autoplay** начинает воспроизведение музыки автоматически – не используйте его! Это делает веб-сайты невыносимыми и раздражающими.

Добавьте код на HTML-страницу в том месте, где вы хотите, чтобы отображался аудиоплеер.

Внедрение изображений-карт

Чтобы продемонстрировать, как создать изображение-карту, мы добавим изображение панели навигации на наш веб-сайт.

Сначала внедрите изображение в подходящее место вашего кода, используя тег <img>. Хорошим место был бы раздел боковой панели:

```
<img src="img/imgmap.png" width="220px" usemap="#foodmenu">
```

Добавьте атрибут usemap и наименование изображения-карты.

Теперь, чтобы создать изображение-карту, используйте теги <map>…</map>. Присвойте карте наименование, используя атрибут name. Оно должно соответствовать атрибуту usemap в теге <img>, который вы добавили ранее:

```
<map name = "foodmenu">
  <area shape= "rect" coords = "0,0,0,0"
      href = "menu.htm">
</map>
```

Внутри тегов <map>…</map> вам необходимо определить горячие точки на частях изображения, по которым пользователь должен щелкнуть мышью.

Вам необходимо создать эти горячие точки, используя систему координат (coords= “x,y,x,y”). Это соответствует координатам x&y в левом верхнем углу и координатам x&y в правом нижнем углу активной точки в пределах размеров изображения.

Чтобы найти эти координаты и размеры изображения, вам понадобится редактор изображений. Вы можете скачать GIMP, который является отличной бесплатной альтернативой Photoshop. В GIMP есть инструмент для измерения пикселей, который весьма полезен для этой задачи.

www.gimp.org/downloads/

Нажмите “Download GIMP directly”.

Отткройте изображение в GIMP. Изображение должно быть того же размера, что и в HTML-документе, поэтому при необходимости вам придется изменить его размер. В этом примере мы установили ширину изображения в нашем HTML-коде равной 220 пикселям.

Наша боковая панель имеет ширину 220 пикселей (x) и длину 598 пикселей (y) (Рис. 6-7).

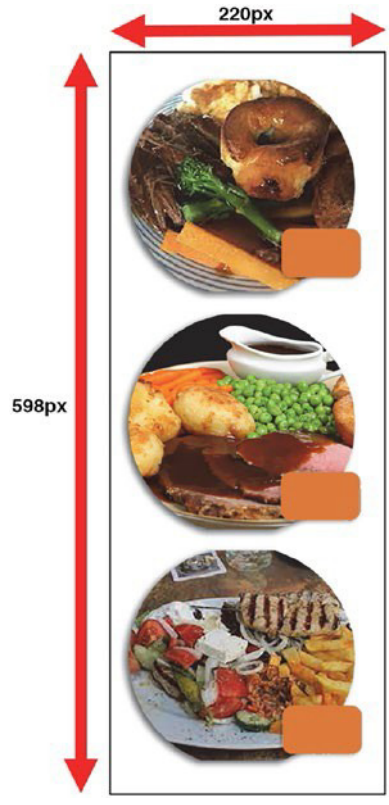


Рис. 6-7. *Изменение размера изображения*

Нам нужно разделить изображение на три части. Первая часть — это первая горячая точка на изображении-карте, вторая — вторая горячая точка, а третья — третья горячая точка.

Чтобы найти их с помощью GIMP, наведите указатель мыши на изображение. Вы увидите набор координат в левом нижнем углу экрана (Рис. 6-8).

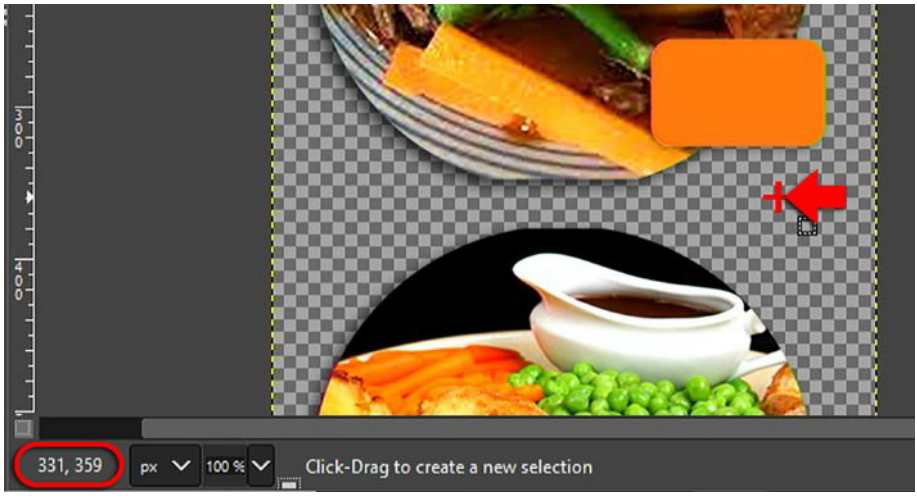


Рис. 6-8. *Координаты в GIMP*

Переместите указатель мыши в правый нижний угол первого фрагмента изображения и обратите внимание на координаты (220, 202) на Рис. 6-9.

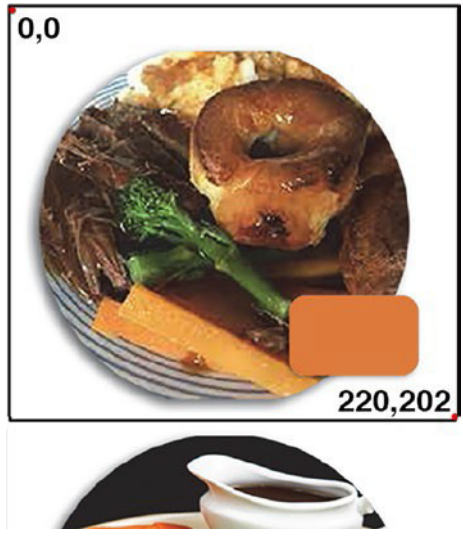


Рис. 6-9. *Координаты первого фрагмента изображения*

Мы можем добавить это значение к изображению-карте:

```
<map name = "foodmenu">
  <area shape= "rect" coords = "0,0,220,202"
      href = "menu.htm">
</map>
```

Сделайте то же самое для двух других фрагментов изображения (Рис. 6-10).

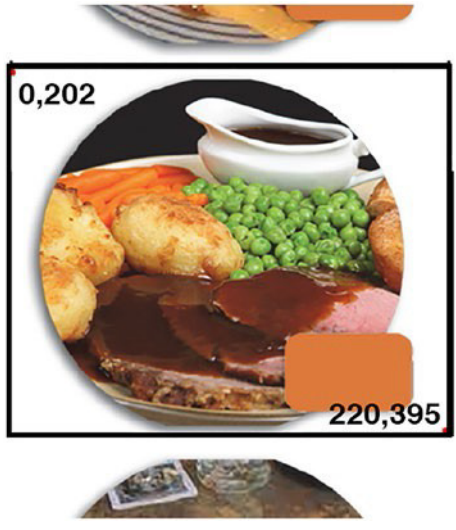


Рис. 6-10. *Координаты фрагмента изображения*

В итоге у вас получится что-то вроде этого:

```
<img src="img/menumap.png" width="220px" usemap="#foodmenu">

<map name = "foodmenu">
  <area shape= "rect" coords = "0,0,220,202" href =
  "menu1.htm">
  <area shape= "rect" coords = "0,202,220,395" href =
  "menu2.htm">
  <area shape= "rect" coords = "0,395,220,596" href =
  "menu3.htm">
```

```
</map>
```

Теперь, когда вы щелкнете на изображении на боковой панели, вы попадете на HTML-страницу, указанную в атрибуте href (Рис. 6-11).

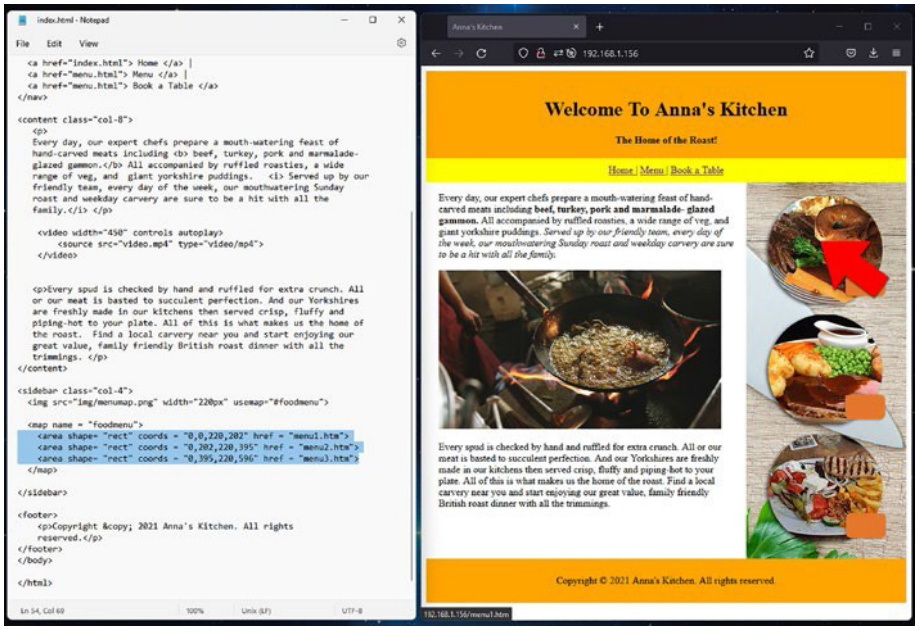


Рис. 6-11. *Окончательный результат*

Упражнение

1. Создайте новый HTML-файл.
2. Где вы храните видео и аудио файлы для внедрения на веб-страницу?
3. Внедрите на веб-страницу видео с YouTube или одно из своих собственных видео.
4. Ниже внедрите аудиозапись в формате MP3.

Заключение

- Используйте тег <video>, чтобы внедрить видео на свою веб-страницу.
- Если вы внедряете видео с YouTube, скопируйте и вставьте код для внедрения из видео YouTube.
- Используйте тег <audio>, чтобы внедрить аудиозапись, например музыку или другой аудиофайл.

ГЛАВА 7

HTML-формы

Форма — это HTML-документ, используемый для сбора вводимых пользователем данных (Рис. 7-1). Информация, введенная пользователем, обычно отправляется на сервер и обрабатывается скриптом.

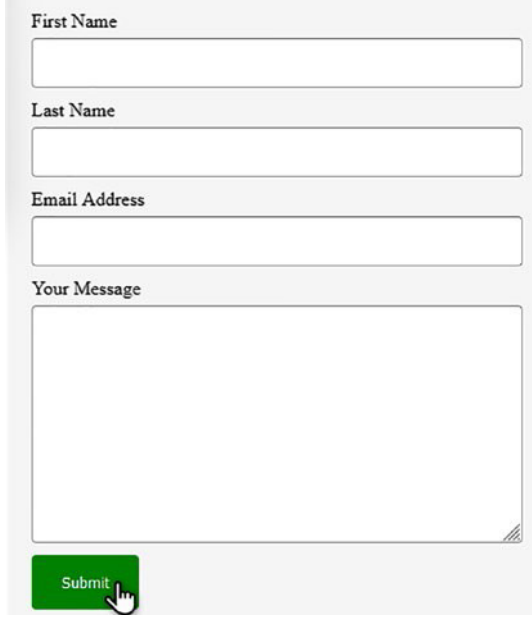


Рис. 7-1. *Форма*

HTML-форма содержит поля для ввода имени, пароля, номера телефона и адреса электронной почты, а также более крупные поля для ввода сообщений. Вы также можете добавить переключатели, поля выбора и кнопку отправки.

Внедрение форм

Формы предоставляют способ получения информации от пользователя.

Используйте теги:

<form action=" ">...</form>

Используйте атрибут действия, чтобы указать на сценарий PHP или CGI для обработки введенных данных.

Внутри тегов <form> вам нужно добавить несколько элементов ввода.

Типы вводимой информации

Типы вводимой информации, позволяющие собирать информацию от пользователя, могут представлять собой

- Текстовые поля, такие как
  - текст
  - телефон
  - e-mail
  - url
  - пароль
- Радиокно́пка
- Флажок
- Ползунок диапазона
- Кнопка
  - отправить
  - сбросить
- Файл

Используйте элемент <input> для определения типа ввода:

```
<input type = " " name = " ">... </input>
```

Атрибут type указывает тип ввода, например текстовое поле, радиокнопка, флажок и т.д.

Атрибут name определяет наименование элемента <input>, на него можно ссылаться в JavaScript, либо для идентификации введенных данных после отправки формы на обработку.

Вы также можете добавить атрибут id, который присваивает идентификатор, позволяющий селектору идентификаторов JavaScript или CSS легко получить доступ к элементу <input>.

Текстовые поля

Текстовое поле представляет собой тип ввода информации и может принимать текст, номера телефонов, адреса электронной почты или пароли. Каждый тип поля предварительно настраивается на определенный формат и тип текста. Например, тип электронной почты предполагает ввод знака @, обозначающего адрес электронной почты. Тип пароля маскирует буквы с помощью *** при их вводе.

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

Текстовая область

Это текстовое поле, в котором допускается размещение нескольких строк текста, его лучше всего использовать при вводе абзацев текста, например сообщения в контактной форме (Рис. 7-2):

```
<textarea name= "message" rows= "5" cols= "30">
    ---
</textarea>
```

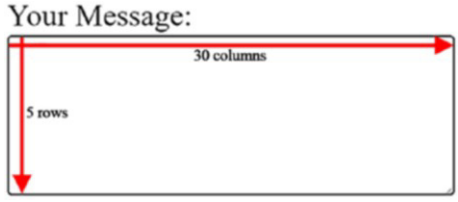


Рис. 7-2. *Сообщение*

Радиокнопки

Позволяют пользователю выбирать нужное из заранее заданных значений:

```
<input type= "radio" name= "gender" value= "female">
    </input>
```

Например, см. Рис. 7-3.

Рис. 7-3. *Радиокнопка*

Флажок

Позволяет создавать группу опций, из которых пользователь может выбирать нужную (Рис. 7-4):

```
<input type="checkbox" name="mainoption1" value="Starter">
<label for=" mainoption1"> I will have a starter</label>
```

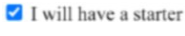


Рис. 7-4. *Флажок*

Список выбора

Позволяет создавать раскрывающийся список заране заданных параметров, из которых пользователь может выбирать нужный (Рис. 7-5):

```
<select>
    <option value= "US">United States</option>
    <option value= "UK">United Kingdom</option>
    <option value= "EU">Europe</option>
</select>
```

Рис. 7-5. *Раскрывающийся список*

Метки

Метки используются для обозначения полей в форме (Рис. 7-6):

```
<label for = "name"> Name: </label>
```

Рис. 7-6. *Метка*

Атрибут “for” должен соответствовать наименованию поля, которое вы помечаете. Например, предыдущая метка обозначает следующее текстовое поле:

```
<input type= "text" name= "name" width= "350"> </input>
```

Кнопка отправки

Кнопка отправки отправляет все значения формы обработчику формы, который обычно представляет собой серверный скрипт:

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

либо

```
<button type="submit">Submit</button>
```

В итоге у вас получится что-то вроде Рис. 7-7.

Рис. 7-7. *Кнопка отправки*

Создание формы

Давайте вернемся к проекту нашего веб-сайта и создадим форму бронирования столиков. Здесь мы добавили код для создания нашей формы.

Код формы располагается между тегами <form> в теле HTML. Внутри тега формы с помощью атрибута действия мы указываем сценарий, который хотим выполнить, когда пользователь нажимает кнопку отправки. Это будет скрипт PHP или Python, который выполняется на сервере (в этом примере мы будем использовать скрипт PHP). Выберем метод отправки для публикации и укажем идентификатор формы. Рассмотрим это позже.

```
<form action="book.php" id="form" method="post">
```

Затем мы добавим метку для поля имени, а затем и само поле ввода. В поле ввода зададим тип (текст, адрес электронной почты или пароль), затем укажем наименование и идентификатор поля:

```
<label for="firstname">First Name</label>
<input type="text" id="firstname" name="firstname">
```

Сделайте то же самое для оставшихся полей.

Идентификаторы будут использованы позже в скрипте Python или PHP, который обрабатывает данные формы.

В итоге мы получим что-то вроде Рис. 7-8.

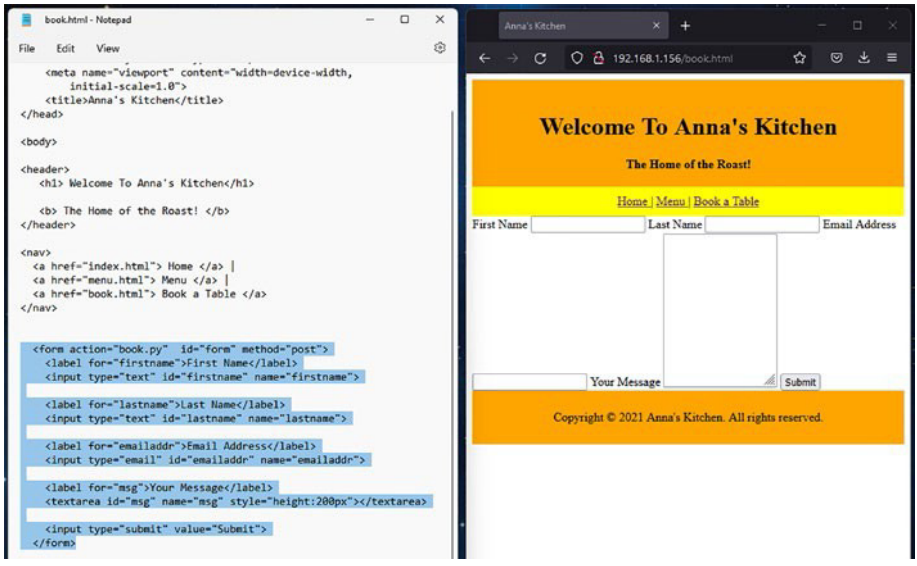


Рис. 7-8. *Форма на веб-сайте*

Стилизация формы

Обратите внимание, что форма выглядит немного грубоватой. Мы можем использовать наши селекторы CSS, чтобы стилизовать форму и сделать ее немного лучше.

Для этого мы можем добавить стили CSS в файл index.css, который мы использовали в предыдущих главах.

Во-первых, нам нужно стилизовать поля ввода. В этой форме у нас есть три разных типа поля ввода.

Ввод в виде обычного текста

input[type=text]

Ввод электронной почты (при этом проверяется действительность адреса электронной почты)

input[type=email]

Большое текстовое поле

textarea

Мы хотим, чтобы все эти три типа имели один и тот же стиль, чтобы мы могли сгруппировать их вместе. Мы хотим, чтобы ширина занимала все окно с отступами в 12 пикселей по краям, чтобы отодвинуть форму от края окна браузера:

```
input[type=text], input[type=email], textarea { width:
  100%;
  padding: 12px;
  border: 1px solid grey; border-
  radius: 4px; margin-top: 5px;
  margin-bottom: 10px;
}
```

Я также добавил тонкую рамку вокруг каждого поля ввода и окрасил ее в серый цвет и закруглил края по углам.

Я добавил 5 пикселей сверху каждого поля (верхнее поле) и 10 пикселей под каждым полем (поле внизу), чтобы разделить их в форме по вертикали.

Далее мы можем стилизовать кнопку отправки. Мы можем сделать это с другим типом ввода:

```
input[type=submit] { background-color: orange; color: white;
  padding: 15px;
  border: none; border-radius: 4px; cursor: pointer;
}
```

Далее мы можем стилизовать кнопку отправки. Мы можем сделать это используя другой тип ввода.

Я также добавил отступ в 15 пикселей вокруг внутренней части кнопки, удалил рамку и добавил 4 пикселя в border-radius, чтобы края кнопки закруглились.

Свойство cursor: pointer превращает указатель мыши в указатель руки, когда вы наводите указатель мыши на кнопку.

Наконец, нам нужно создать контейнер для хранения формы. Мы можем сделать это, используя тег в HTML-коде:

```
<div  class="formcontainer">
```

Мы можем стилизовать контейнер формы в коде CSS, как мы это делали ранее:

```
.f ormcontainer  {
  background-color:  whitesmoke;
  padding:  20px;
}
```

Здесь я создал контейнер с серым фоном и отступом в 20 пикселей вокруг формы, чтобы отодвинуть ее от остального содержимого страницы.

Как только мы это сделаем, у нас получится что-то вроде Рис. 7-9. Обратите внимание, что форма выглядит намного лучше.

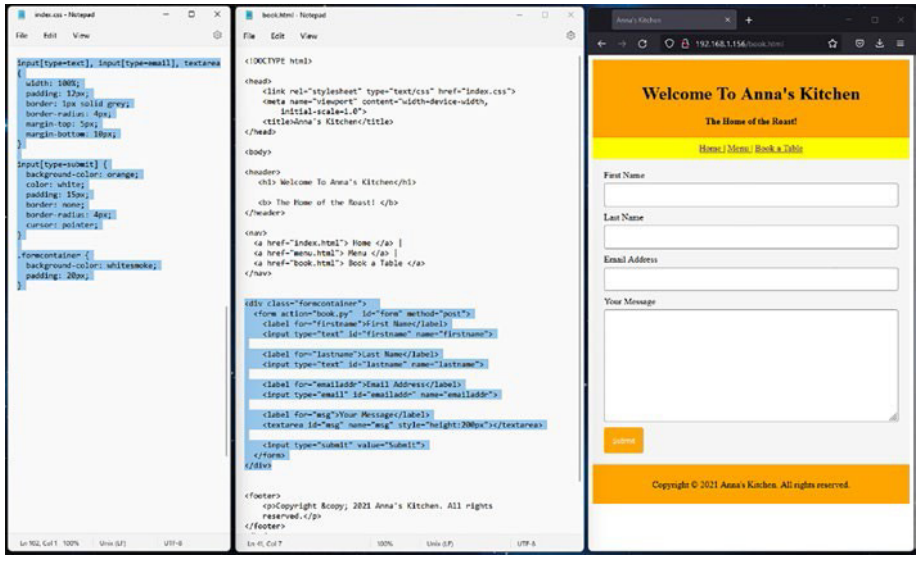


Рис. 7-9. *Отформатированная форма*

Обработка данных формы

Далее нам нужно добавить в форму функциональность (т. е. то, что происходит, когда вы нажимаете кнопку submit).

Форма обычно обрабатывается на сервере с помощью сценария PHP или Python, принимает введенные данные и отправляет их на адрес электронной почты. В этом примере мы собираемся использовать для обработки данных формы PHP-скрипт.

Настройка веб-сервера для выполнения сценариев

Теперь, чтобы это заработало, вам понадобится поддержка PHP, установленная на вашем веб-хосте (за подробностями обратитесь к своему хостинг-провайдеру), или, если вы используете веб-сервер Abyss, который мы установили в Главе 1, вам необходимо установить PHP-скриптинг. Для этого перейдите на следующий

сайт:

aprelium.com/downloads/

В случае Windows загрузите предварительно настроенный пакет PHP для Windows (64-разрядной версии) (Рис 7-10).

Рис. 7-10. *Установка PHP*

Перейдите в папку загрузок, затем дважды щелкните PHP8xx-x64.exe. Запустите настройку и нажмите «Next», чтобы начать процесс (Рис. 7-11).

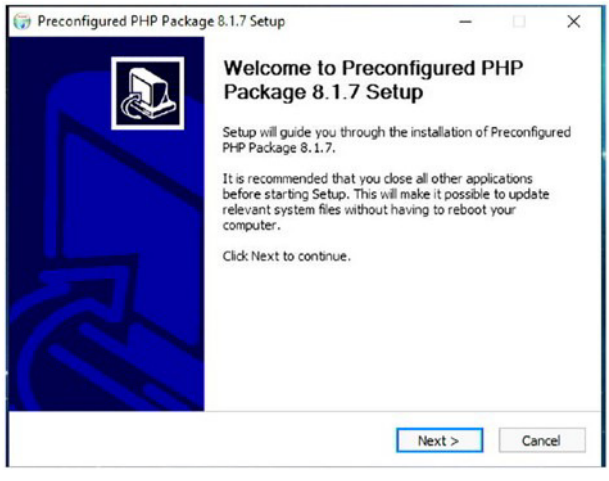


Рис. 7-11. *Установка PHP*

Примите лицензионное соглашение и нажмите «Install». Введите путь, куда будет установлено программное обеспечение (Рис. 7-12), в

нашем случае C:\Program Files\PHP8.

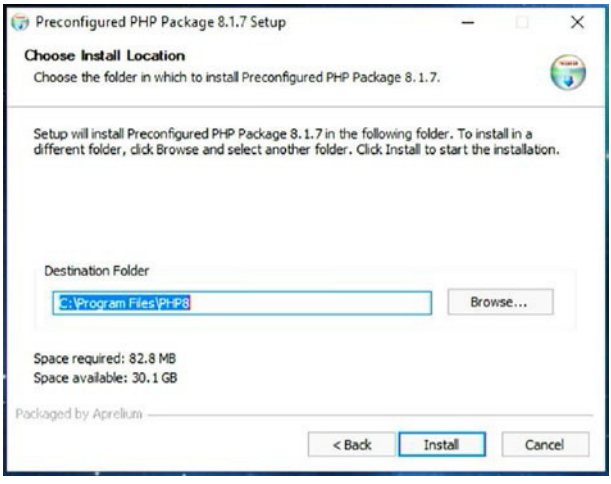


Рис. 7-12. *Выбор места установки*

Затем откройте веб-браузер и перейдите к консоли сервера. Введите пароль администратора сервера.

127.0.0.1:9999

В таблице хостов нажмите «Configure» (Рис. 7-13).

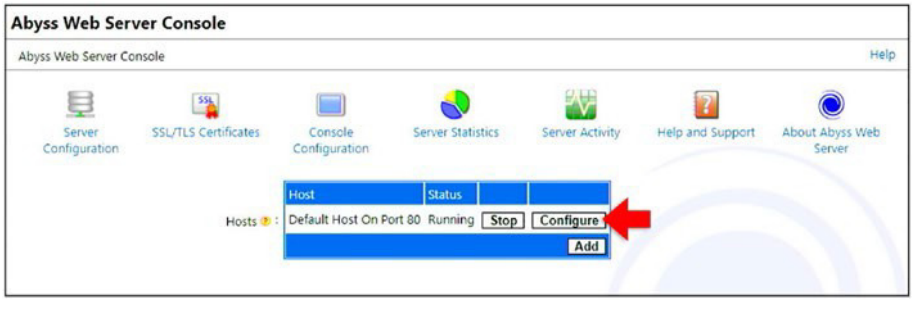


Рис. 7-13. *Конфигурирование консоли*

Выберите Scripting Parameters (Рис. 7-14).

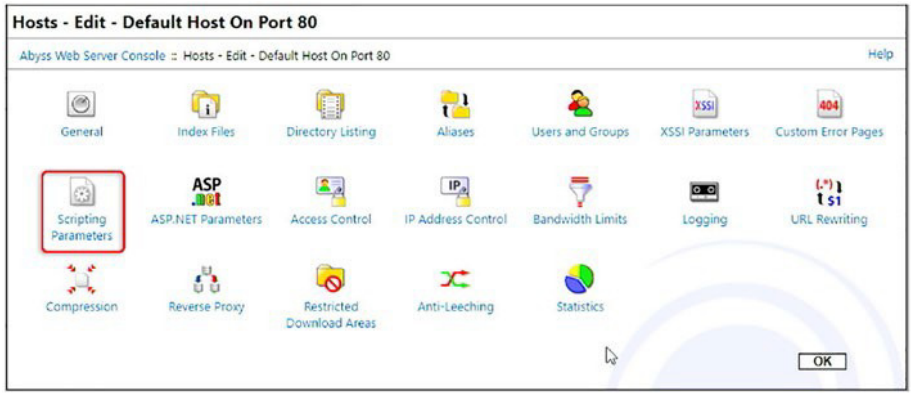


Рис. 7-14. *Scripting Parameters*

Установите флажок Enable Scripts Execution, затем нажмите Add в таблице Interpreters (Рис. 7-15).

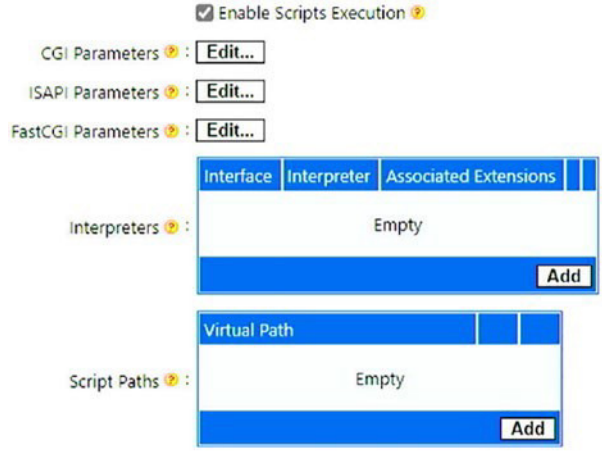


Рис. 7-15. *Таблица Interpreters*

Установите для интерфейса значение FastCGI (локальный — каналы). В поле «Interpreter» нажмите «Browse», затем перейдите в каталог, в который вы установили PHP 8 (Рис. 7-16). Выберите php-cgi.exe. Установите стандартный тип, затем установите флажок «Use the associated extensions to automatically update the script paths».

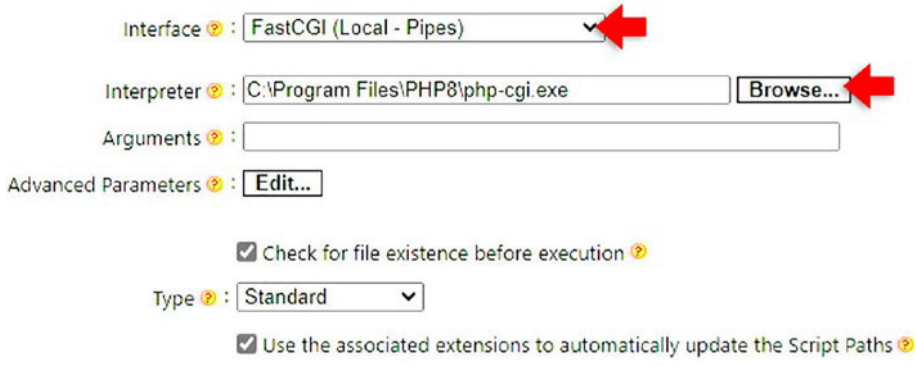


Рис. 7-16. *Конфигурирование*

Нажмите «Add» в таблице «Associated Extensions», затем введите php в поле расширения. Нажмите «OK», затем еще раз «OK» (Рис. 7-17).

Рис. 7-17. *Таблица Associated Extensions*

Теперь нам нужно добавить расширение php в индексные файлы. Вернувшись на главный экран, нажмите «Index Files» (Рис. 7-18).

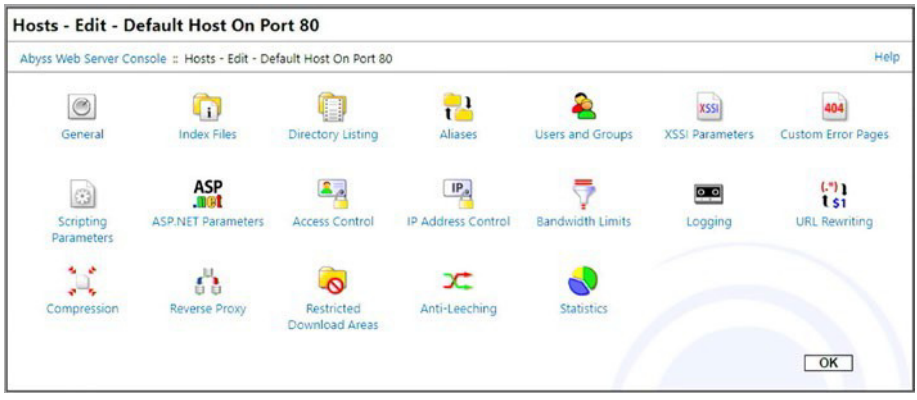


Рис. 7-18. *Выбор Index Files*

Нажмите «Add» в правом нижнем углу раздела «Index Files» (Рис. 7-19).

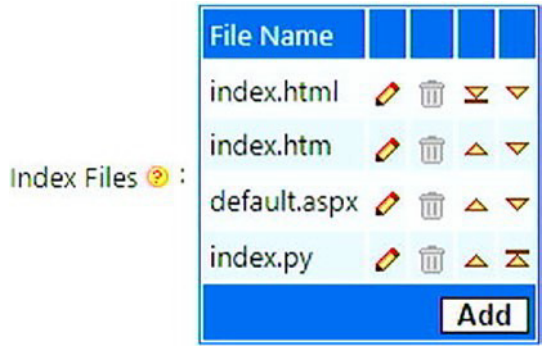


Рис. 7-19. *Раздел Index Files*

Введите index.php и нажмите «OK». Затем еще раз нажмите «OK» (Рис. 7-20).

Рис. 7-20. *Ввод index.php*

Вернувшись на главный экран, нажмите «Restart» (Рис. 7-21).

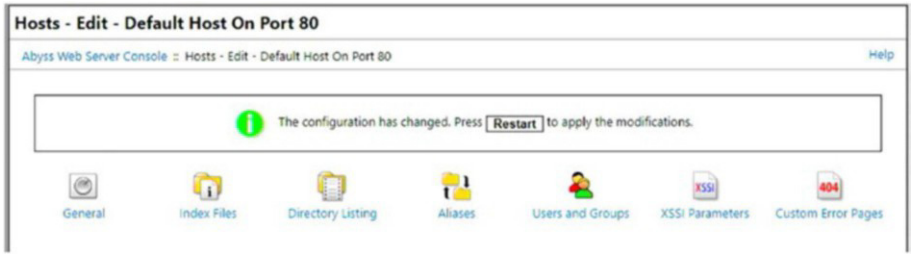


Рис. 7-21. *Нажать Restart*

Выполнение сценария

Теперь, когда мы нажмем кнопку отправки, сервер выполняет PHP-скрипт под названием book.php (Рис. 7-22).

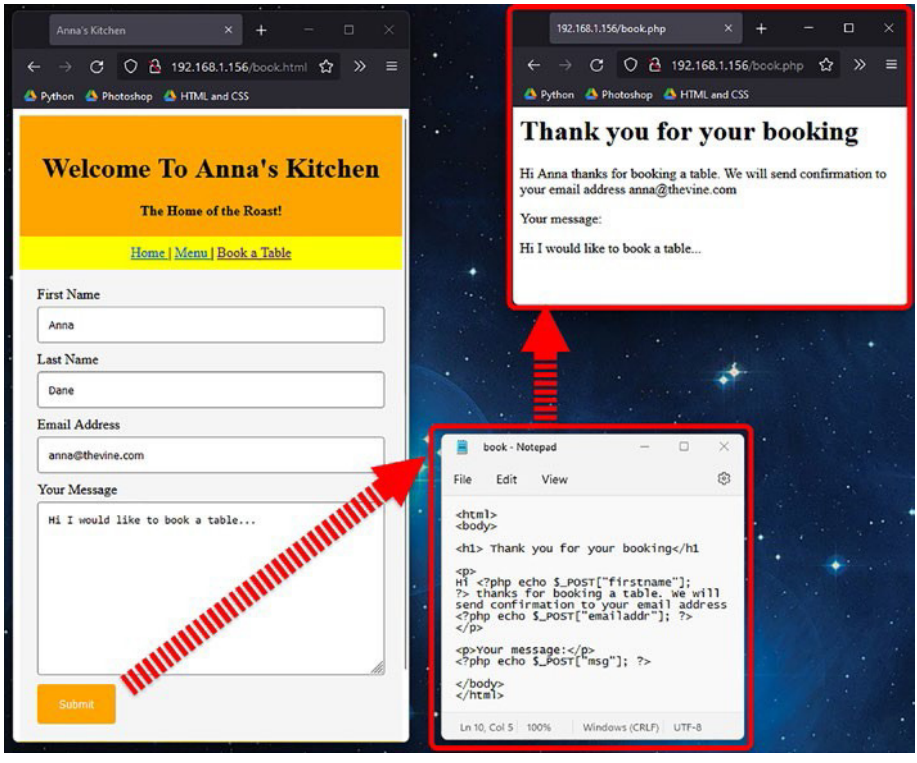


Рис. 7-22. *Скрипт, выполняемый после нажатия кнопки «Submit»*

Я внедрил следующим образом пример PHP-скрипта под названием book.php, который вы можете использовать, чтобы опробовать форму:

```
<html>
<head>
    <title> Thanks for your booking</title>
</head>
<body>

<h1> Thank you for your booking</h1>
    <p>Hi <?php echo $_POST["firstname"]; ?>
```

```
    thanks for booking a table. We will send confirmation to
    your email address
    <?php echo $_POST["emailaddr"]; ?></p>
    <p>Your message:</p>
    <?php echo $_POST["msg"]; ?>

</body>
</html>
```

Метод отправки

Существует два метода отправки данных формы на сервер: get и post.

В этой демонстрации у нас есть PHP-скрипт, который выполняется, когда пользователь нажимает кнопку «Submit», и отправляет данные, введенные в поля формы, обратно на сервер. Сначала мы выполним скрипт посредством метода «get», затем выполним тот же скрипт посредством метода «post», чтобы вы могли увидеть различия в инспекторе заголовков, показанном в правом нижнем углу скриншотов.

Get

Метод «get» добавляет введенные данные к запрашивающему URL-адресу, разделенному знаком «?» (Рис. 7-23). Это называется строкой запроса.

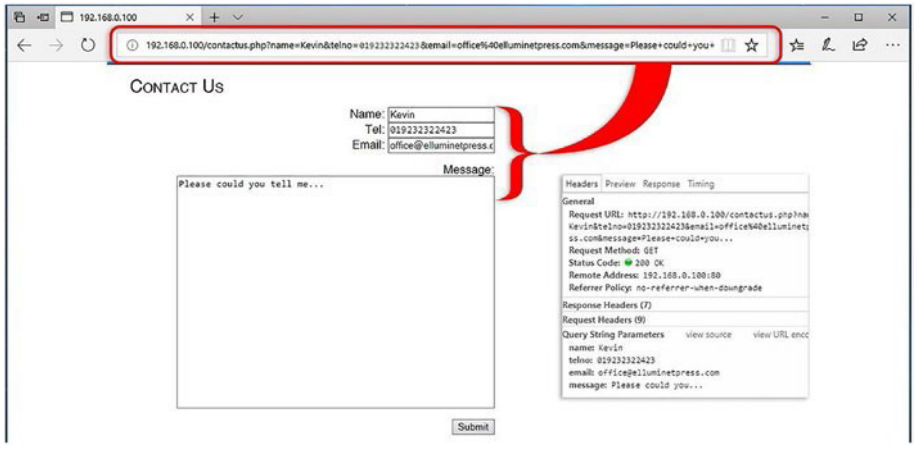


Рис. 7-23. *Строка запроса*

Вы можете увидеть данные, добавленные к URL-адресу сценария на предыдущем рисунке, а атрибуты строки запроса в инспекторе заголовков.

Этот метод никогда не следует использовать для отправки конфиденциальной информации, такой как пароли, поскольку он четко виден в URL-адресе страницы.

Post

При использовании метода «post» все данные передаются с помощью HTTP-заголовков сценария обработки, а не через URL-адрес. Обратите внимание, что URL-адрес в адресной строке вверху браузера чистый (Рис. 7-24).

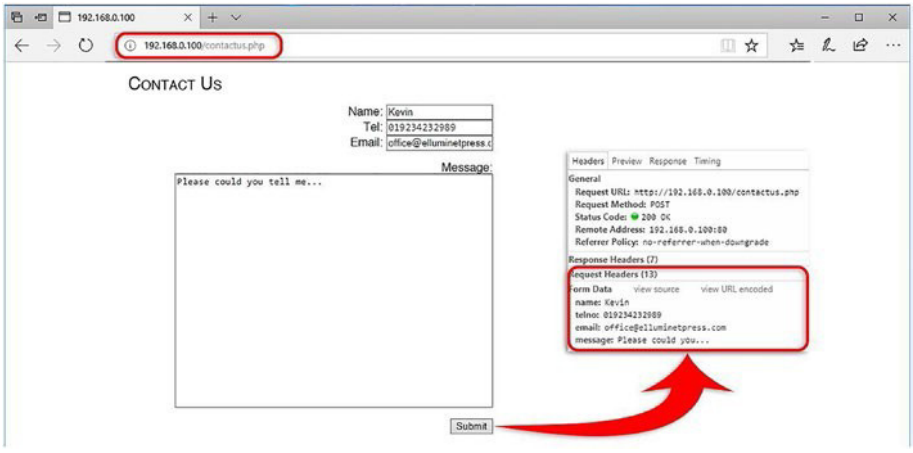


Рис. 7-24. *Использование метода Post*

Вы можете увидеть данные, добавленные к заголовку скрипта на предыдущей иллюстрации.

Упражнение

1. Создайте HTML-форму для приема некоторой информации от пользователя, такой как имя, адрес электронной почты и т.д.
2. Добавьте немного стиля, чтобы форма выглядела немного лучше.
3. Какие методы отправки используются для передачи на сервер данных, введенных в форму? Что вам следует использовать?
4. Что такое строка запроса?
5. Как обрабатываются данные формы?

Заключение

- Используйте <form action=“ ”>…</form>, чтобы создать форму и указать сценарий для обработки данных.
- Метод отправки «get» добавляет введенные данные к запрашивающему URL-адресу.
- Метод «post» отправляет данные с HTTP-заголовками сценария обработки, а не через URL-адрес.

ГЛАВА 8

Введение в JavaScript

Первоначально названный LiveScript, JavaScript был разработан Бренданом Эйхом из Netscape в середине 1990-х годов. Позже в 1995 году он был переименован в JavaScript.

JavaScript — это интерпретируемый объектно-ориентированный язык сценариев на стороне клиента и который позволяет создавать динамический контент на веб-странице. Вы можете анимировать изображения, проверять данные и создавать интерактивные элементы. Другими словами, JavaScript добавляет активность к веб-страницам и поддерживается большинством современных веб-браузеров, таких как Chrome, Firefox, Edge и Safari.

Есть три способа добавить JavaScript на веб-страницу:

1. Вы можете встроить код между тегами <script> в свой HTML-документ. Вы можете добавить это в раздел <head> или <body> вашего HTML-документа. В примере на Рис. 8-1 мы добавили код JavaScript между тегами <script> в заголовке.

```
<!DOCTYPE html>
<html>

<head>
    <title>Welcome to JavaScript</title>

    <script>
        function mult(a, b) {
        document.getElementById("desc").innerHTML = a * b;
        }
    </script>

</head>

<body>
   <h1>Welcome to JavaScript</h1>

   <p id="desc"></p>

   <button type="button" onclick="mult(2, 2)">Multiply</button>
</body>

</html>
```

Рис. 8-1. *Внедрение JavaScript*

2. Вы можете сохранить код JavaScript в отдельном файле и связать его со своим HTML. Это полезно для более крупных проектов, в которых у вас есть несколько HTML-документов, использующих одни и те же функции JavaScript.

   В этом примере код JavaScript сохраняется в файле script.js. Мы включаем файл JavaScript, используя атрибут src в открывающем теге <script>. Мы можем добавить эту строку в заголовок HTML-файла (Рис. 8-2).

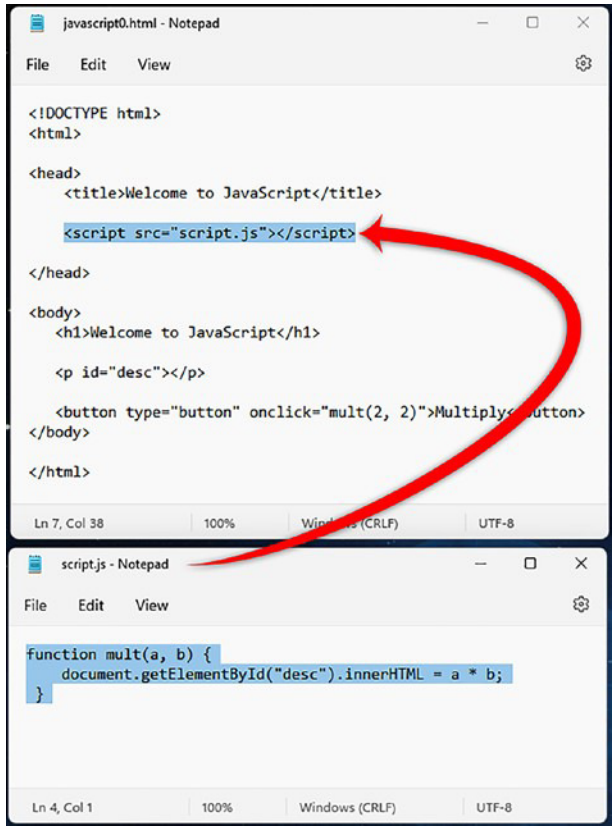


Рис. 8-2. *Внедрение JavaScript*

3. Вы также можете разместить код JavaScript непосредственно внутри тега HTML. Это известно как встроенный код.

    Click Here

Синтаксис JavaScript

Программа JavaScript, называемая сценарием, представляет собой список операторов, содержащих конструкции и команды, выполняющие действия. Давайте взглянем на некоторые основные строительные блоки программы JavaScript.

Операторы

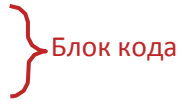
Оператор — это фрагмент кода, который выполняет действие и является основным строительным элементом программы JavaScript. Используйте точку с запятой, чтобы указать конец оператора:

```
document.getElementById("desc").innerHTML = "This is a test";
```

Блоки

Блок — это последовательность операторов, которые часто выполняются вместе и обычно используются в конструкциях управления потоком, таких как оператор if или в циклах while и for. Блок группируется парой фигурных скобок { }:

```
if (some condition) {
   statement 1;
   statement 2;
   statement 3;
}
```



Идентификаторы

Идентификатор — это наименование, которое вы выбираете для переменных, параметров и функций. Здесь мы объявили переменную с именем firstNum:

```
let firstNum = 5;
```

Идентификаторы также определяют наименования функций. Здесь у нас есть новая функция под названием addNum:

```
function addNum(n1, n2) {
  return n1 + n2;
}
```

Ключевые слова

При использовании JavaScript существуют зарезервированные Ключевые слова, которые имеют определенное назначение (Рис. 8-3). По этой причине вы не можете использовать ключевые слова в качестве идентификаторов или имен свойств.

abstract	delete	function	null	throw
boolean	do	goto	package	throws
break	double	if	private	transient
byte	else	implements	protected	true
case	enum	import	public	try
catch	export	in	return	typeof
char	extends	instanceof	short	var
class	false	int	static	void
const	final	interface	super	volatile
continue	finally	long	switch	while
debugger	float	native	synchronized	with
default	for	new	this	

Рис. 8-3. *Ключевые слова JavaScript*

Комментарии

Комментарии игнорируются при выполнении программы; однако вы всегда должны комментировать свой код, чтобы другие могли понять его функцию. Вы можете добавлять комментарии, заключая комментарий между /* и */.

```
/* This is a multiline comment and
      is often used to describe a section of
      code. */
```

Если вы хотите прокомментировать только одну строку, используйте //.

```
// this is a single-line comment
```

Первая программа

Давайте посмотрим на пример. Вот простая HTML-страница. Мы внедрили код JavaScript в теги <script>:

```
<!DOCTYPE html>
<html>

<body>
   <h1>Welcome to JavaScript</h1>

   <p id="desc"></p>

   <script>
      document.getElementById("desc").innerHTML = "This is a
      test";
   </script>
```

```
</body>

</html>
```

Давайте внедрим код JavaScript в HTML-документ (Рис. 8-4). Вверху мы добавили заголовок. Мы также добавили код JavaScript между тегами <script> в теле HTML-документа.

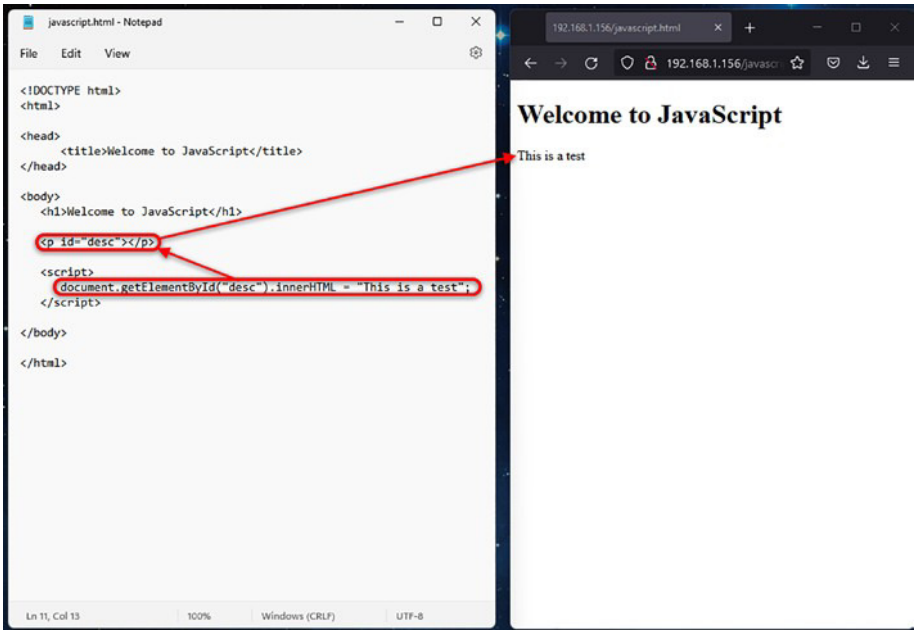


Рис. 8-4. *Внедрение кода JavaScript в HTML*

Здесь мы видим, что метод getElementById() присваивает указанный нами текст объекту <p> в документе. Вот как мы можем использовать JavaScript для управления HTML-документом.

Метод getElementById отображает вывод текста в окно браузера. «Ну и что, — скажете вы, — HTML уже это делает». Да, это так, но с помощью JavaScript вы можете делать то, что невозможно сделать с помощью простого HTML. Например, вы можете отображать текст или параметры на основе переменных или определенных условий. Вы

также можете использовать JavaScript для добавления интерактивности и проверки контента на статический веб-сайт.

Лабораторная работа

В этом упражнении мы создадим небольшую HTML-форму с функцией JavaScript, которая складывает два числа и отображает результат.

Сначала нам нужно создать форму. Обратите внимание, что мы не используем обработчик отправки формы, поскольку нам не нужно использовать внешний скрипт для обработки данных, мы просто хотим выполнить функцию JavaScript:

```
<form>
```

Далее нам нужно создать поля. Мы добавим метку, а затем поле типа ввод номера — метка с идентификатором firstnum и поле с идентификатором Secondnum. Эти идентификаторы будут использоваться функцией JavaScript позже. Тег
 просто переносит текст на новую строку:

```
<label for="firstnum">First Number: </label>
<input type="number" id="num1"><br>
<label for="secondnum">Second Number: </label>
<input type="number" id="num2"><br><br>
```

Теперь нам нужен элемент HTML, который мы можем использовать для отображения результата. Элемент span — это универсальный контейнер для встроенного контента:

```
<span id="res"></span> <br>
```

Затем мы добавляем кнопку, которую пользователь может нажать, чтобы добавить значения:

```
<input type="button" value="Add"
```

Теперь, когда пользователь нажимает кнопку, возникает событие onclick. Нам нужно сообщить событию onclick, что наша функция должна запускаться, когда пользователь нажимает кнопку. Итак, в строке, которую мы добавили ранее, мы можем добавить событие onclick и вызвать функцию:

```
<input type="button" value="Add" onclick="add();">
```

В итоге у нас получится что-то вроде Рис. 8-5.

Рис. 8-5. *HTML-форма с функцией JavaScript*

После того, как мы создали HTML-страницу, нам нужно добавить JavaScript. В этом случае я собираюсь внедрить JavaScript на настоящую HTML-страницу. Мы делаем это с помощью тегов <script>.

Внутри тегов скрипта сначала мы объявляем нашу функцию сложения чисел:

```
<script>
   function add () {
```

Далее нам нужно создать пару переменных для хранения чисел, чтобы мы могли с ними работать. Для этого нам нужно получить число, которое было введено в поля формы (num1 и num2). Мы можем использовать метод getElementById для получения значения из num1 и num2 (Рис. 8-6).

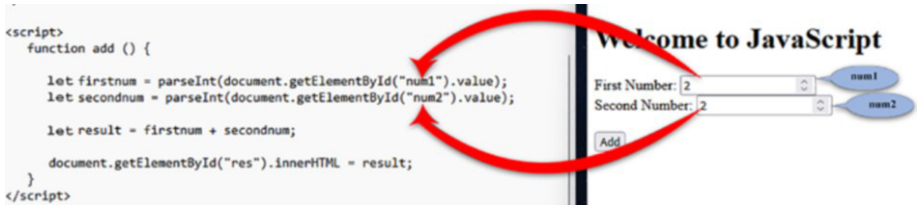


Рис. 8-6. *Получение чисел*

Поскольку значения, введенные в форму, представляют собой текст (строку), нам необходимо преобразовать их в целое число (или целые числа). Для этого мы используем функцию parseInt(). После преобразования эти значения присваиваются переменным firstnum и secondnum:

```
let firstnum = parseInt(document.getElementById("num1").
value);
let secondnum = parseInt(document.getElementById("num2").
value);
```

Далее нам нужно сложить два числа и присвоить ответ переменной result:

```
let result = firstnum + secondnum;
```

Мы можем записать значение результата в элемент HTML-диапазона с идентификатором «res»:

```
document.getElementById("res").innerHTML = result;
```

Здесь мы видим, что значение результата присваивается значению HTML элемента span на веб-странице (Рис. 8-7).

Рис. 8-7. *Значение, присвоенное HTML*

Давайте рассмотрим, как работает код, когда мы открываем его в браузере. Если вы вводите два числа в поля, после нажатия кнопки вызывается функция add(). Первые две строки функции получают значения из полей формы (например, 2). Затем значения преобразуются в целые числа (с помощью parseInt()) и присваиваются значениям firstnum и secondnum соответственно. Затем значения суммируются, и ответ присваивается результату (Рис. 8-8).

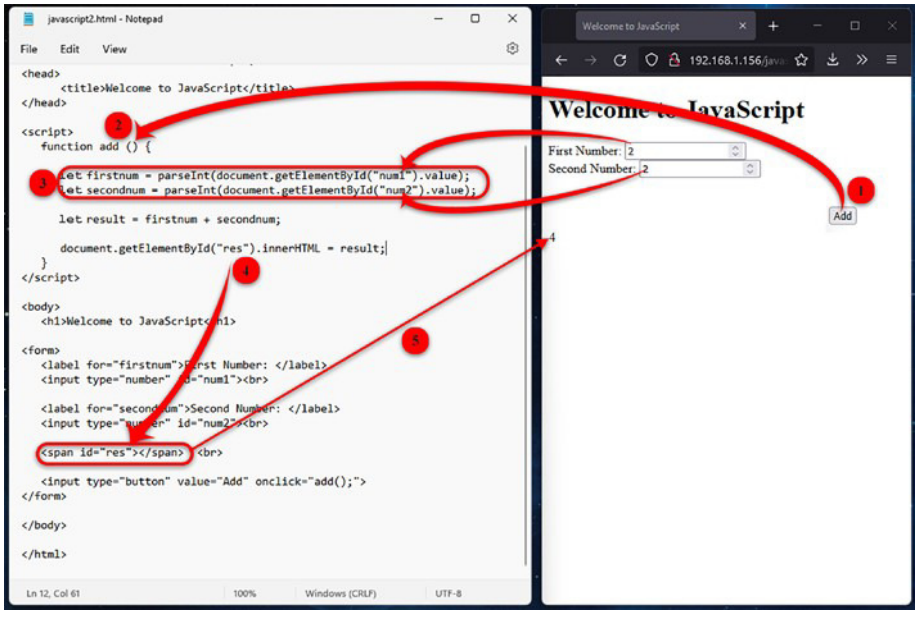


Рис. 8-8. *Значения, присвоенные HTML*

Значение результата затем присваивается значению HTML элемента span на странице HTML. Это то, что отображает результат (в данном примере «4»).

Попробуйте это.

Заключение

- Внедряйте код JavaScript между тегами <script>…</script>.
- Включите внешний файл сценария, используя атрибут src в открывающем теге <script>.

  <script src="…"></script>

- Точка с запятой отмечает конец оператора.
- querySelector
- querySelectorAll
- getElementById

ГЛАВА 9

Системы управления контентом

Система управления контентом (или CMS) — это приложение, которое позволяет создавать, редактировать и хранить цифровой контент. Другими словами, CMS — это часть программного обеспечения, которая работает на веб-сервере и позволяет легко создавать, редактировать и управлять контентом, опубликованным на веб-сайте.

Данные сайта хранятся в базе данных. Платформа CMS берет на себя все технические аспекты создания и управления веб-сайтом. Конечный пользователь может использовать текстовый редактор WYSIWYG, похожий на текстовый процессор, для создания, публикации и редактирования контента без необходимости опыта программирования. Тем не менее, базовые знания HTML и CSS являются преимуществом.

Большинство платформ CMS поставляются с набором готовых шаблонов, называемых темами, которые вы можете использовать для быстрой настройки внешнего вида вашего сайта. Вы также можете загрузить бесчисленное множество других тем или даже разработать свои собственные.

В настоящее время это наиболее распространенный способ создания веб-сайта вместо использования статических HTML-страниц.

WordPress на сегодняшний день является самой популярной системой управления контентом, на ней работает примерно 43%

веб-сайтов в Интернете; однако есть и другие, такие как Drupal, Joomla и Umbraco. Более подробную информацию об использовании этих платформ вы можете найти на следующих сайтах:

- wordpress.org
- drupal.org
- joomla.org
- umbraco.com

Вместо создания статических HTML-страниц содержимое веб-сайта хранится в базе данных (обычно MySQL). Страницы, которые вы видите при посещении веб-сайта, генерируются динамически и форматируются в соответствии с используемой темой.

Используя CMS, такую как WordPress, веб-сайты можно создавать очень быстро, обладая минимальными техническими знаниями или навыками программирования.

Разработчики могут разрабатывать плагины и темы для веб-сайта, а редакторы и издатели могут входить в систему и редактировать контент без необходимости иметь дело с кодом, используя интерфейс, похожий на текстовый процессор,.

На Рис. 9-1 мы видим редактор на стороне сервера веб-сайта WordPress. Пользователь может входить в систему и добавлять страницы и сообщения в блогах на сайте без необходимости использования какого-либо кода.

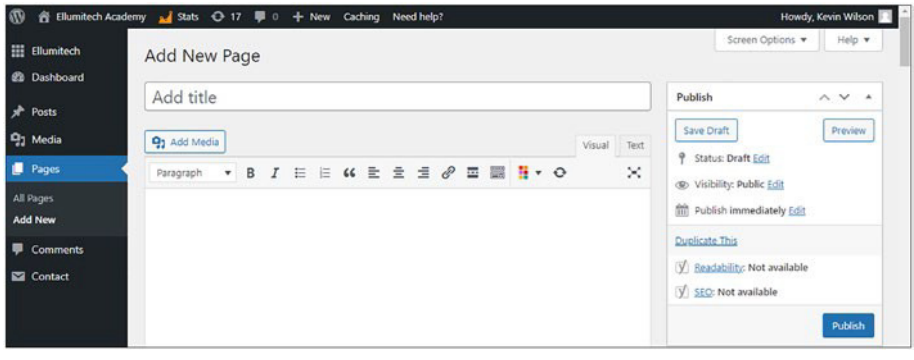


Рис. 9-1. *Серверная часть WordPress*

Когда кто-то посещает веб-сайт, он видит опубликованный контент (Рис. 9-2). Это называется клиентской частью.

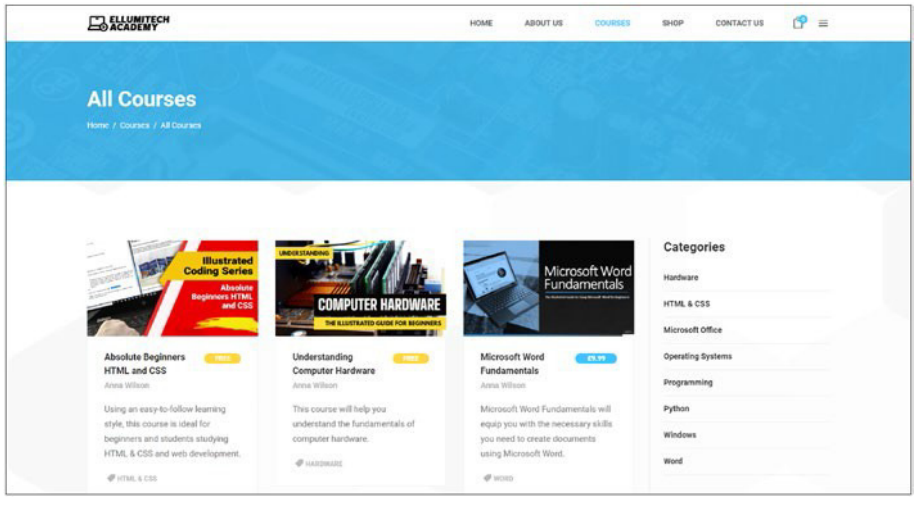


Рис. 9-2. *Клиентская часть WordPress*

Установка WordPress на нашем сервере

Если вы хотите поэкспериментировать с WordPress, вы можете загрузить его со следующего сайта:

wordpress.org/download/

Нажмите “Download WordPress”. Это приведет к загрузке zip-файла в ваш каталог загрузок.

Вам потребуется установить PHP и настроить его, как мы это делали в разделе «Настройка веб-сервера для выполнения сценариев» Главы 7.

Далее установите MySQL. Для этого скачайте установщик с сайта MySQL:

dev.mysql.com/downloads/installer/

На странице загрузок рядом с “Windows (x86, 32-bit), MSI Installer, 5.5M” нажмите “Download” (Рис. 9-3).

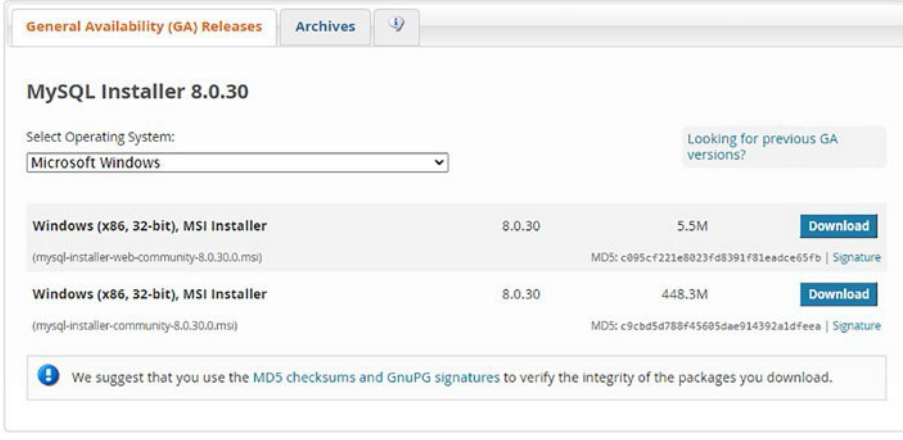


Рис. 9-3. *Страница установки MySQL*

Запустите только что скачанный установщик. Вы найдете его в папке загрузок. Пройдите процесс настройки. Когда вы перейдете на экран “Setup Type”, выберите “Server only”, нажмите “Next”, а затем “Execute” (Рис. 9-4).

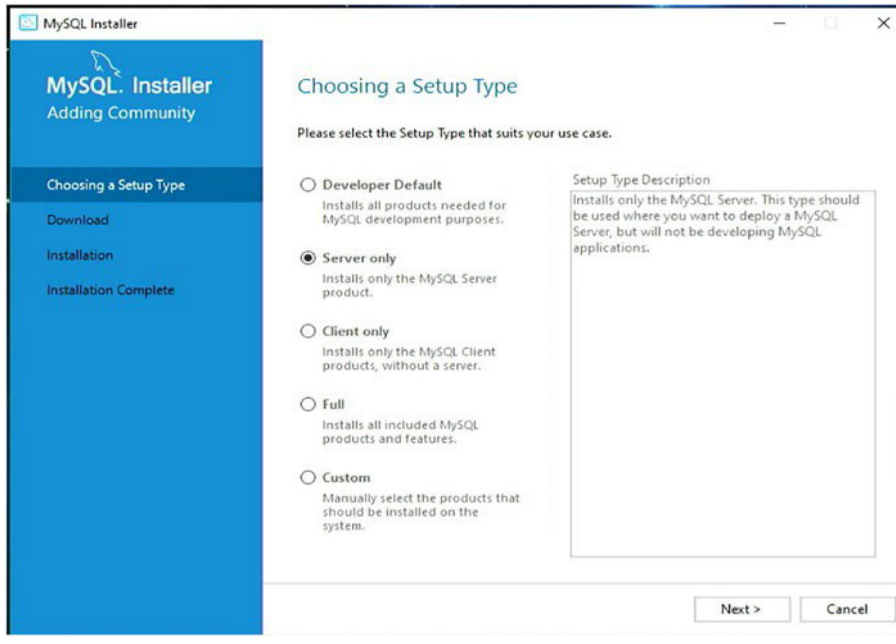


Рис. 9-4. *Меню Setup Type*

Выберите “Development Computer” в списке “Type and Networking” (Рис. 9-5). Нажмите “Next.”

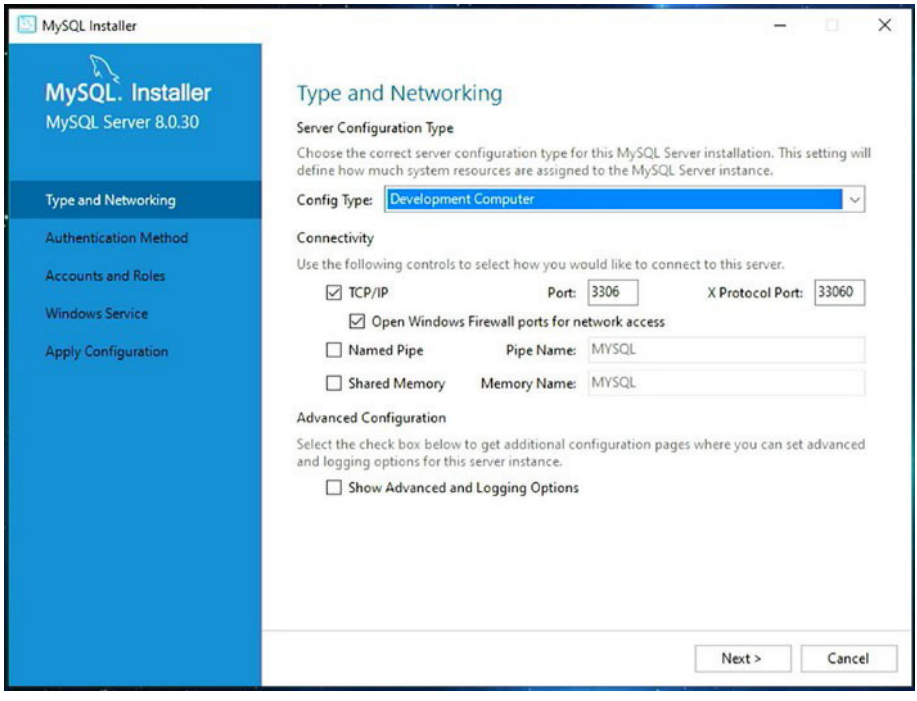


Рис. 9-5. *Меню опций Type and Networking*

На экране “Accounts and Roles” введите root-пароль (Рис. 9-6). Это пароль, который вы будете использовать для создания и администрирования своих баз данных, поэтому не забывайте его. Нажмите “Next”, чтобы продолжить.

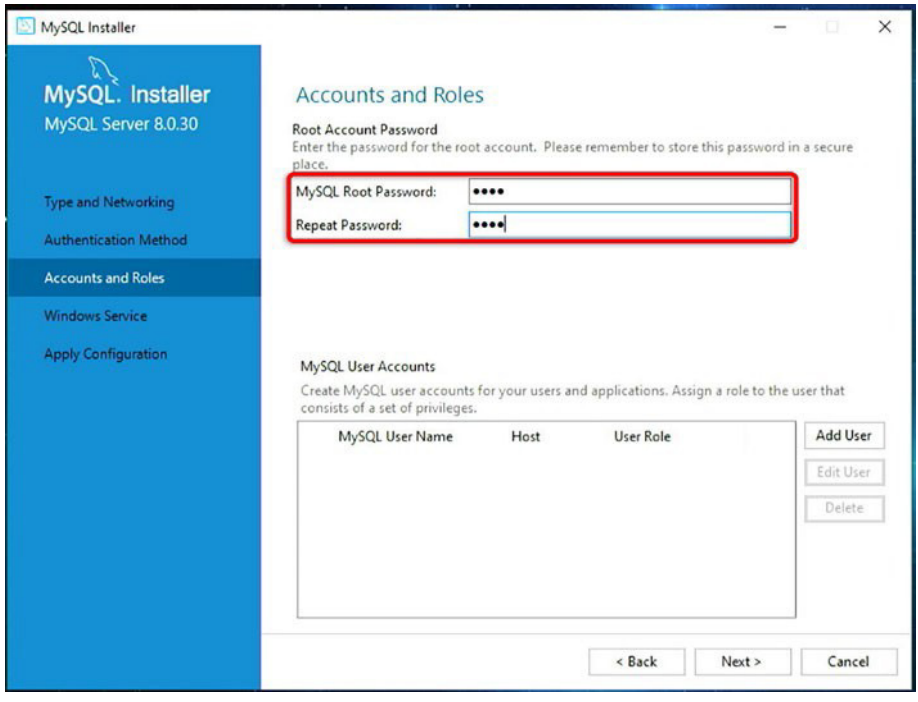


Рис. 9-6. *Ввод Root-пароля*

Нажмите “Next” на экране “Windows Service” (Рис. 9-7). Вы можете оставить их такими, каими они принимаются по умолчанию.

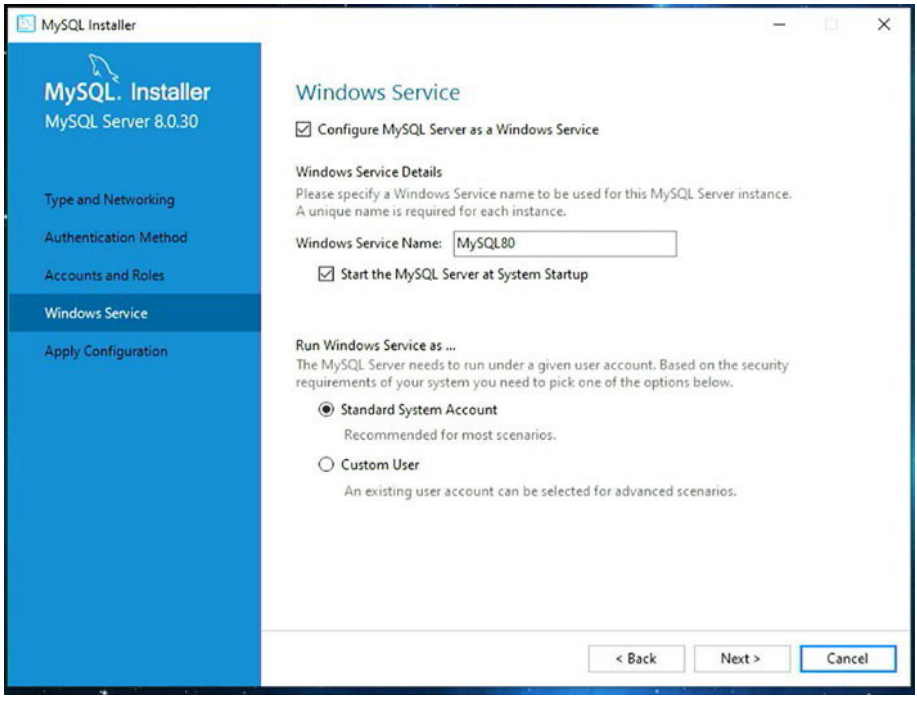


Рис. 9-7. *Меню Windows Service*

Затем нажмите “Execute”, чтобы начать установку (Рис. 9-8).

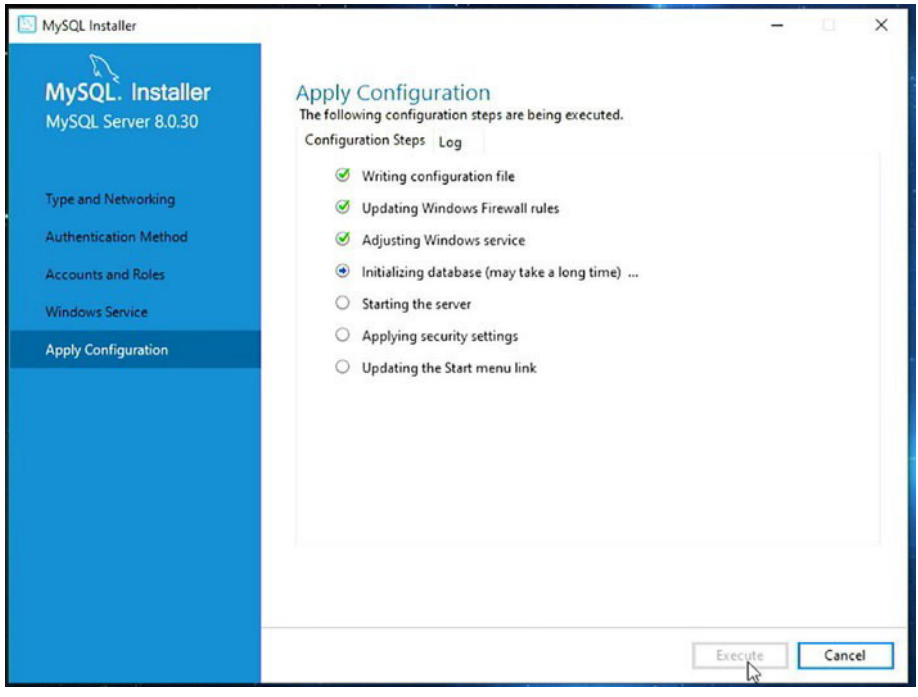


Рис. 9-8. *Меню Apply Configuration*

Разрешить установку программного обеспечения.

Теперь давайте создадим нашу базу данных. В меню «Пуск» прокрутите вниз до «MySQL», затем нажмите “MySQL Command Line Client” (Рис. 9-9).

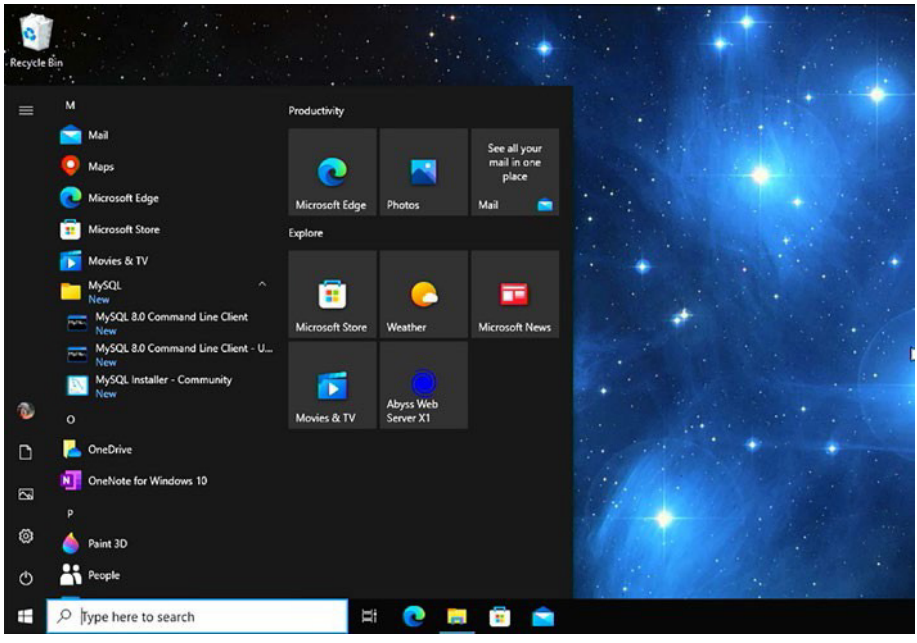


Рис. 9-9. *Клиент командной строки MySQL в меню «Пуск»*

Введите root-пароль, который вы выбрали при установке сервера MySQL (Рис. 9-10).

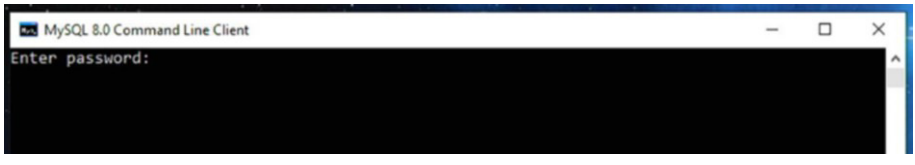


Рис. 9-10. *Ввод Root-пароля*

Теперь нам нужно задать имя пользователя базы данных, которое сайт WordPress будет использовать для подключения к базе данных (Рис. 9-11).

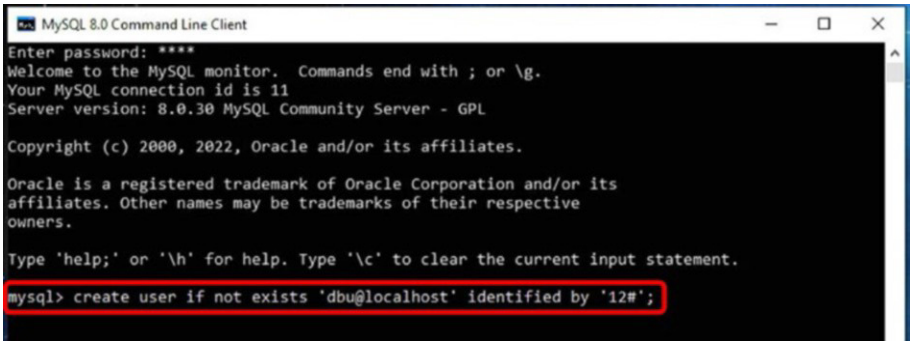


Рис. 9-11. *Клиент командной строки*

Введите следующее в командной строке MySQL:

```
create user if not exists 'dbu@localhost' identified by '12#'
```

Это создаст пользователя с именем dbu и паролем 12#. Это простой пример для демонстрации, но вместо этого вам следует использовать надежный пароль.

Теперь нам нужно создать базу данных. Введите следующее в командной строке MySQL:

```
Create database if not exists 'mydb';
```

Это создаст новую базу данных под названием mydb (Рис. 9-12). Опять же, вам следует задавать для вашей базы данных осмысленное имя.

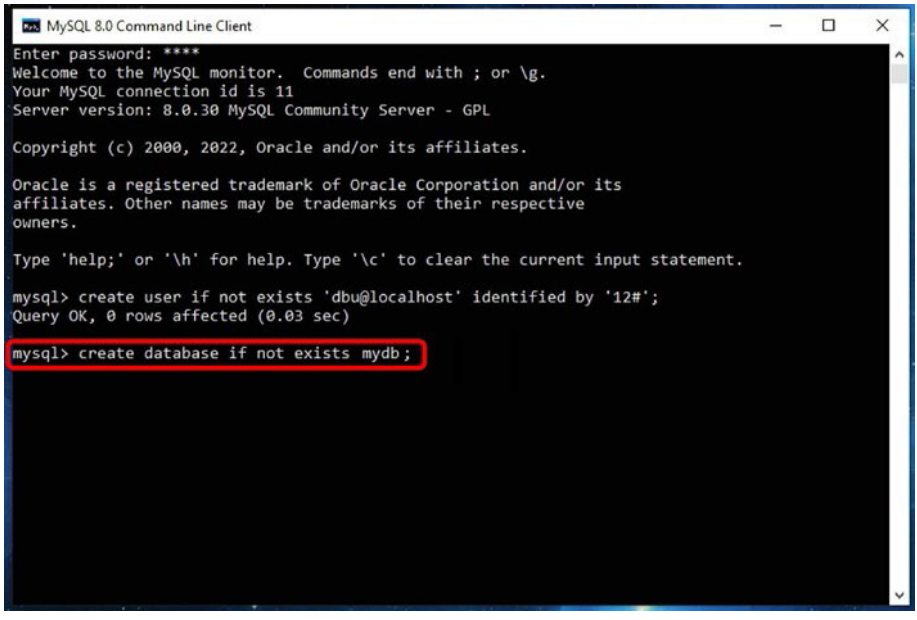


Рис. 9-12. *Создание базы данных*

Наконец, нам нужно дать разрешение пользователю dbu, который мы создали ранее. Введите следующую строку:

grant all privileges on mydb.* to ‘dbu@hostname ’;

Здесь мы добавляем все привилегии, необходимые для WordPress, нашему пользователю dbu, чтобы он имел возможность доступа и использования mydb.

Далее нам нужно установить WordPress.

Откройте проводник, перейдите в папку загрузок, щелкните правой кнопкой мыши zip-файл WordPress, который вы скачали ранее, и выберите «Извлечь все» во всплывающем меню (Рис. 9-13).

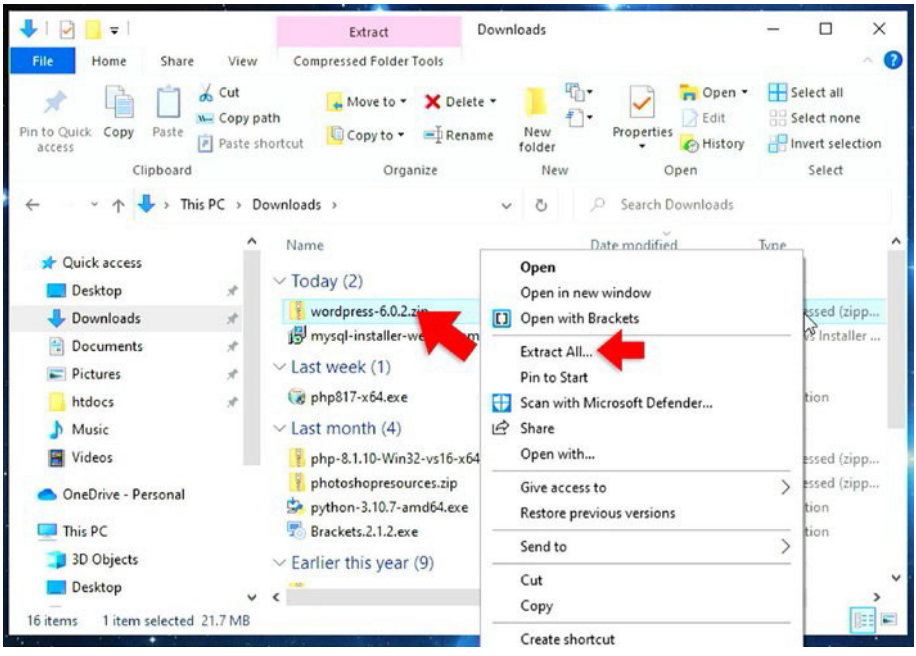


Рис. 9-13. *Извлечние WordPress из ZIP-архива*

Укажите каталог, в котором будут сохранены файлы htdocs вашего веб-сервера. При использовании Abyss это будет C:\Abyss Web Server\htdocs (Рис. 9-14).

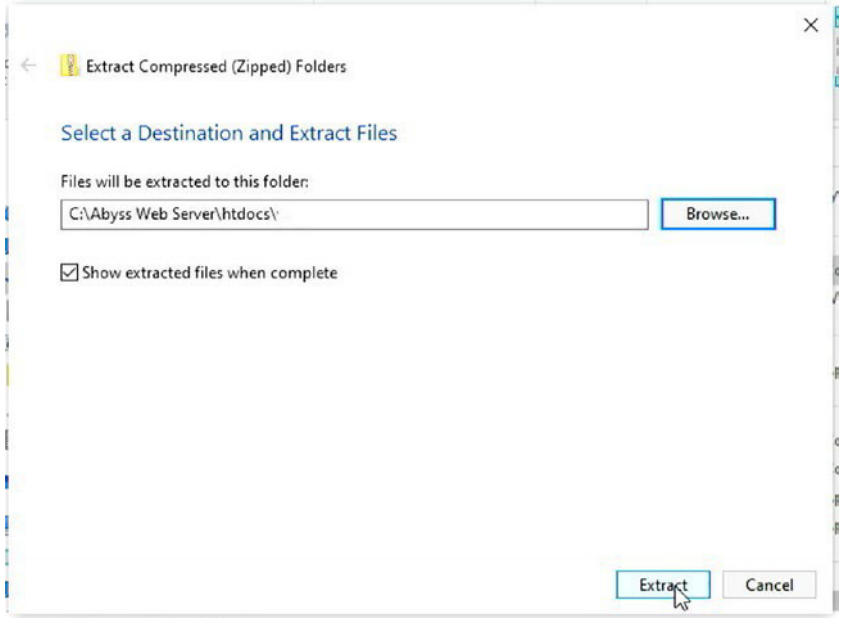


Рис. 9-14. *Выбор места на веб-сервере*

После извлечения файлов откройте веб-браузер и перейдите по адресу

127.0.0.1/wordpress

Либо, если не получится, попробуйте

127.0.0.1/wordpress/wp-admin/setup-config.php

Пройдите процедуру настройки. Выберите язык, прокрутите вниз и нажмите “Continue” (Рис. 9-15).

Рис. 9-15. *Выбор языка*

Убедитесь, что у вас есть наименование базы данных, имя пользователя и пароль, которые вы создали ранее. В этом примере это будут

Наименование базы данных: mydb
Пользователь: dbu
Пароль 12#

Обратите внимание: не используйте простые пароли на действующем веб-сайте, используйте что-то более надежное. Я просто упростил пароль, чтобы его было легче понять.

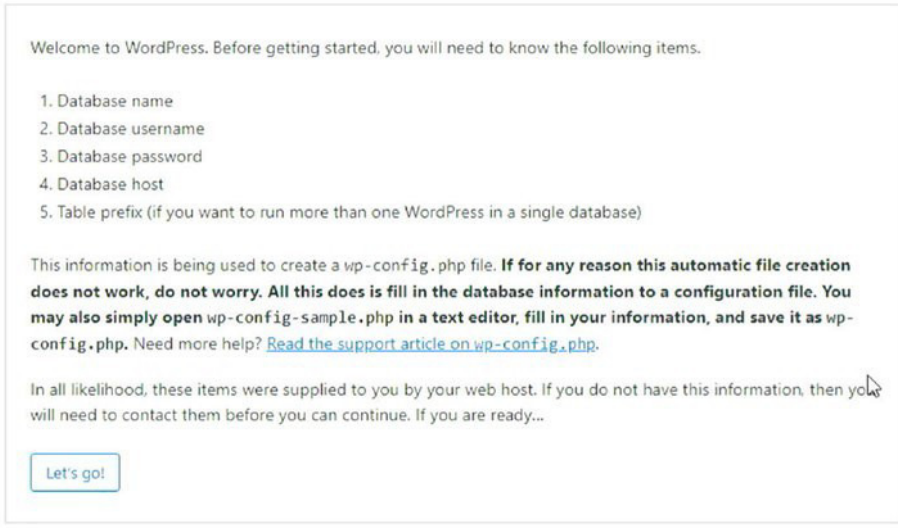


Рис. 9-16. *Установка WordPress*

Нажмите “Let’s go!” (Рис. 9-16).

Введите данные в поля (Рис. 9-17).

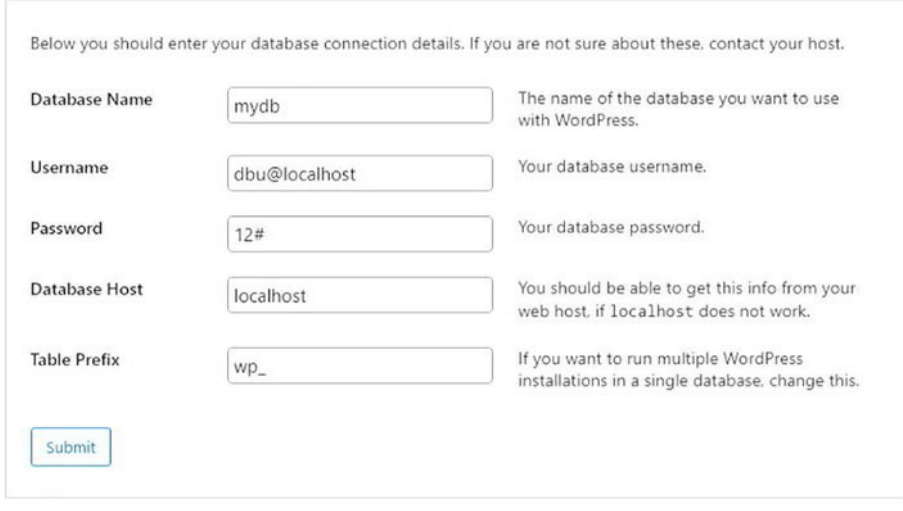


Рис. 9-17. *Заполнение полей*

Нажмите “Submit”.

Введите имя пользователя и пароль администратора. Это данные, которые вы будете использовать для входа в WordPress для создания и редактирования вашего веб-сайта (Рис. 9-18). Нажмите “Install WordPress”.

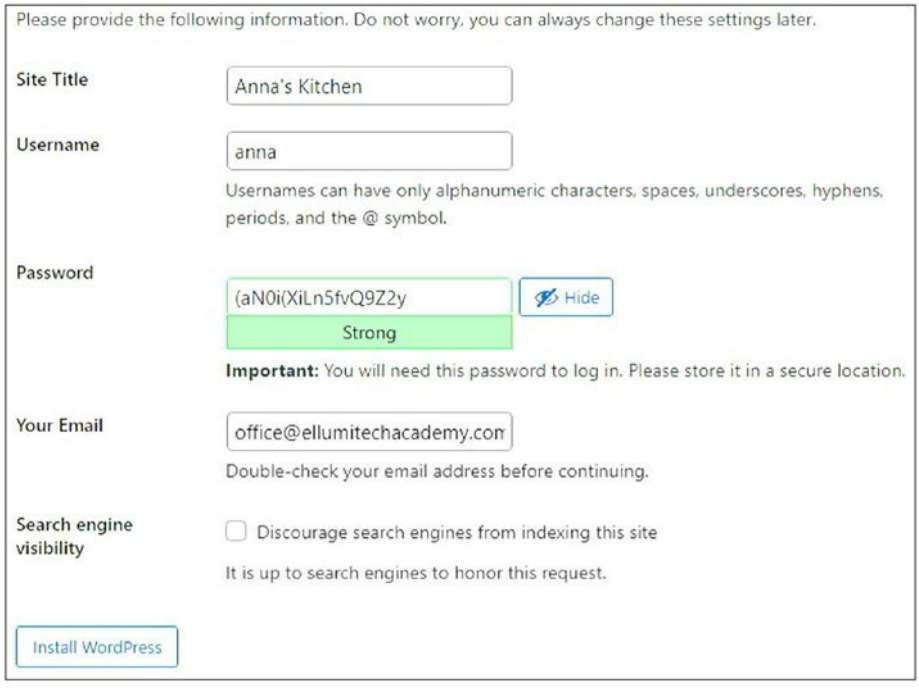


Рис. 9-18. *Конфигурирование информации администратора*

После установки WordPress нажмите “Log In” (Рис. 9-19).

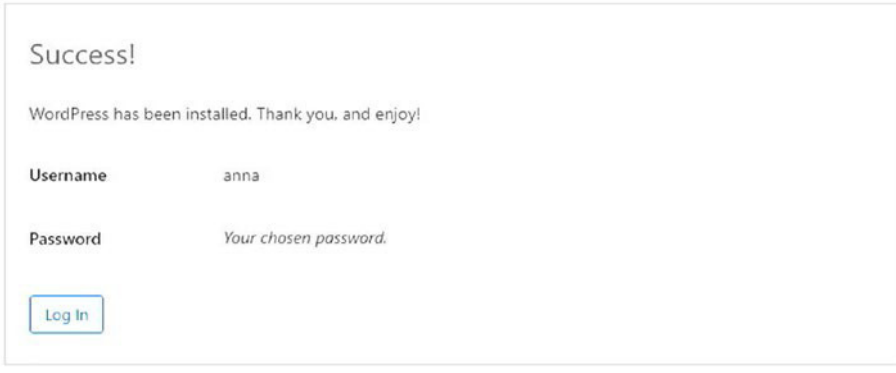


Рис. 9-19. *Вход*

Введите имя пользователя и пароль, которые вы создали при установке WordPress (Рис. 9-20).

Рис. 9-20. *Введите свое имя пользователя и пароль*

Вы попадете на панель управления wp-admin (Рис. 9-21).

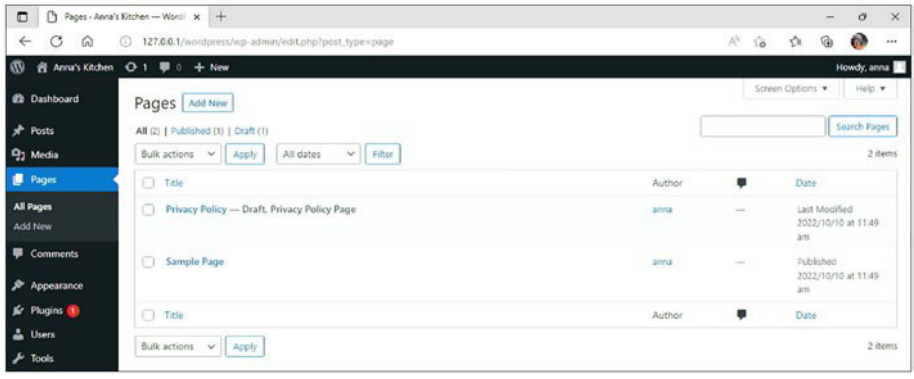


Рис. 9-21. *Панель управления Wp-Admin*

Вы можете получить доступ к своему сайту по адресу

127.0.0.1/wordpress

К администрированию серверной части

127.0.0.1/wordpress/wp-admin

Взгляните на WordPress и посмотрите, как он работает. Попробуйте несколько тем на вкладке внешнего вида слева (Рис. 9-22).

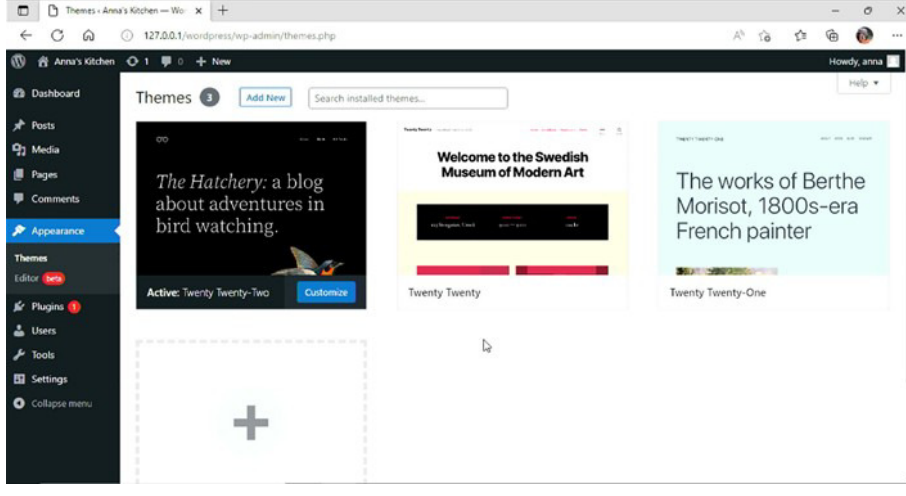


Рис. 9-22. *Варианты внешнего вида*

Попробуйте добавить несколько страниц, используя вкладку страниц.

Фреймворки веб-разработки

Веб-фреймворк — это программная платформа для разработки веб-приложений и веб-сайтов. Такие платформы предлагают широкий спектр предварительно написанных компонентов, фрагментов кода и шаблонов приложений, которые можно использовать для разработки веб-сервисов и других веб-ресурсов.

Существует два типа фреймворков — клиентская (внешняя часть) и серверная (внутренняя часть).

Интерфейсные платформы, такие как React и Angular, используются для разработки пользовательского интерфейса веб-сайта или приложения — той части, которую вы можете увидеть при посещении сайта. Интерфейсные фреймворки в основном построены на JavaScript, HTML и CSS.

Серверные платформы, такие как Flask и ASP.NET, используются для разработки скрытой части приложения или веб-сайта, которая отвечает за управление базой данных, безопасность, маршрутизацию URL-адресов и генерацию страниц. Эти платформы построены на Python, .NET, Ruby, Java и PHP.

Другой фреймворк, который одновременно является и интерфейсной, и серверной частью, — это Django, известный фреймворк Python, используемый разработчиками и предприятиями.

Для получения дополнительной информации об этих платформах посетите веб-сайты, перечисленные в Таблице 9-1.

Таблица 9-1. *Веб-сайты фреймворков*

Фреймворк	Веб-сайт
Django	www.djangoproject.com
Flask	flask.palletsprojects.com
aSp.net	dotnet.microsoft.com
angular	angular.io
react	reactjs.org

Заключение

- Система управления контентом (или CMS) — это приложение, которое позволяет создавать, редактировать и хранить цифровой контент.
- WordPress — безусловно, самая популярная система управления контентом, на которой работает примерно 43% веб-сайтов в Интернете; однако есть и другие, такие как Drupal, Joomla и Umbraco.
- Используя CMS, такую как WordPress, веб-сайты можно создавать очень быстро, обладая минимумом технических знаний или опыта программирования. Разработчики могут создавать плагины и темы.
- Вы можете загрузить его с сайта wordpress.org/download/.
- Вы можете загрузить MySQL с сайта mysql.com/downloads/windows/installer/.
- Веб-фреймворк – это программная платформа для разработки веб-приложений и веб-сайтов.

- Интерфейсные платформы, такие как React и Angular, используются для разработки пользовательского интерфейса веб-сайта или приложения.
- Внутренние платформы, такие как Flask и ASP.NET, используются для разработки серверной части приложения или веб-сайта, которая отвечает за управление базой данных, безопасность, маршрутизацию URL-адресов и генерацию страниц.

ПРИЛОЖЕНИЕ А

Справочник по элементам HTML

A

<!-- -->	Этот тег используется для добавления комментария в HTML-документ	
<!DOCTYPE>	Этот тег используется для указания версии HTML	<!DOCTYPE html>
<a>	Называется тегом привязки и создает гиперссылку или ссылку	 link
<abbr>	Определяет сокращение для фразы или более длинного слова	<abbr title=“full”> abbreviation </abbr>
<area>	Определяет область карты изображения	<area shape=“rect” coords=“ ” href=“ ”>
<article>	Определяет автономный контент	<article> … </article>
<aside>	Определяет контент помимо основного контента. В основном представлен в виде боковой панели.	<aside> … </aside>
<audio>	Используется для встраивания звукового контента в HTML-документ.	<audio controls> <source src=“ ” type=“audio/mpeg”> </audio>

	Используется для выделения текста	…
<blockquote>	Используется для определения контента, взятого из другого источника	<blockquote cite=“ ”> Quote… </blockquote>
<body>	Используется для определения раздела тела HTML-документа	<body> …</body>
 	Используется для применения одного разрыва строки	A line of text
<button>	Используется для обозначения нажимаемой кнопки	<button type = “”> button, reset, submit

C

<canvas>	Используется для предоставления графического пространства в веб-документе	<canvas> … </canvas>
<caption>	Используется для определения заголовка таблицы	<caption> … </caption>
	Используется для выравнивания содержимого по центру	…
<cite>	Используется для определения названия произведения, книги, веб-сайта и т.д.	<cite> … </cite>
<code>	Используется для отображения части программного кода в HTML-документе	<code> … </code>

D

<datalist>	Используется для предоставления предопределенного списка параметров ввода	<datalist id=" "> <option value=" "> … </datalist>
<dd>	Используется для предоставления определения/описания термина в списке описаний	<dd> … </dd>
<del>	Определяет текст, который был удален из документа	<del> … </del>
<details>	Определяет дополнительные сведения, которые пользователь может просмотреть или скрыть	<details> … </details>
<dfn>	Используется для обозначения термина, который определен в предложении/фразе	<dfn> … </dfn>
<dialog>	Определяет диалоговое окно или другие интерактивные компоненты	<dialog open> … </dialog>
	Определяет главу или раздел в HTML-документе	…
<dl>	Используется для определения списка описаний	<dl> … </dl>
<dt>	Используется для определения термина в списке описаний	<dt> … </dt>

E

	Используется для выделения содержимого, применяемого в этом элементе	…
<embed>	Используется как встроенный контейнер для типов text/html, image/jpg, video/mp4	<embed type = “” src = “” width=“”height=“”>

F

<fieldset>	Используется для группировки связанных элементов/меток в веб-форме	<fieldset> <legend> … </legend> </fieldset>
<figcaption>	Используется для добавления подписи или пояснения к элементу <figure>	<figcaption> … </figcaption>
<figure>	Используется для определения автономного контента и чаще всего упоминается как единое целое	<figure> … </figure>
<footer>	Определяет нижний колонтитул веб-страницы	<footer> … </footer>
<form>	Используется для определения HTML-формы	<form action = “” method = “post”> … </form>

H

<h1> to <h6>	Определяет заголовки HTML-документа уровней 1–6
<head>	Определяет заголовок HTML-документа
<header>	Определяет заголовок раздела или веб-страницы
<hr>	Используется для применения тематического разрыва между элементами уровня абзаца
<html>	Представляет корень HTML-документа

I

	Используется для представления текста другим голосом	…
<iframe>	Определяет встроенный фрейм, в который можно вставлять другой контент	<iframe src=“ ” title=“ ”> … </iframe>
<img>	Используется для вставки изображения в HTML-документ	<img src=“ ” width = “ ” height = “ ”>
<input>	Определяет поле ввода в форме HTML. Типы включают текст, телефон, электронную почту, пароль, кнопку, отправку, флажок, радио и т. д.	<input type=“text” id=“ ” name=“ ”>
<ins>	Представляет текст, вставленный в HTML-документ	<ins> … </ins>

K

<kbd>	Используется для определения ввода с клавиатуры	<kbd> … </kbd>

L

<label>	Определяет текстовую метку для поля ввода формы	<label for = “ ”> … </label>
	Используется для представления элементов в списке	…
<link>	Ссылки на внешний ресурс, например файл CSS	<link rel=“stylesheet” type=“text/css” href=“ ” />

M

<main>	Представляет основное содержимое HTML-документа	<main> … </main>
<map>	Определяет карту изображения с активными областями	<map name=“ ”> <area shape=“rect” coords=“ ” href=“ ”> </map>
<mark>	Представляет выделенный текст	<mark> … </mark>
<meta>	Определяет метаданные имени HTML-документа = описание, Ключевое слово, автор, область просмотра	<meta charset=“UTf-8”> <meta name=“ ” content=“ ”>
<meter>	Определяет скалярное измерение с известным диапазоном или дробным значением	<meter id=“ ” value=“ ” min=“ ” max=“ ”> … </meter>

N

<nav>	Представляет раздел страницы для представления	<nav> … </nav>
<noscript>	Предоставляет альтернативный контент, если тип сценария не поддерживается браузером	<noscript> JavaScript not supported! </noscript>

O

<object>	Используется для встраивания объекта в HTML-файл	<object data=“ ” width=“ ” height=“ ”> </object>
	Определяет упорядоченный список элементов	…
<optgroup>	Используется для группировки параметров раскрывающегося списка	<optgroup label=“ ”> <option value=“ ”> … </option> <option value=“ ”> … </option> </optgroup>

P

	Представляет абзац в HTML-документе	…
<pre>	Определяет предварительно отформатированный текст в документе HTML	<pre> … </pre>
<progress>	Определяет ход выполнения задачи в HTML-документе	<progress id=“ ” value=“ ” max=“ ”> … </progress>

Q

<q>	Определяет короткую встроенную цитату <q> … </q>

S

<s>	Отображает текст, который больше не является правильным или актуальным	<s> … </s>
<samp>	Используется для представления примера вывода компьютерной программы	<samp> … </samp>
<script>	Используется для объявления JavaScript в документе HTML	<script> … </script>
<section>	Определяет общий раздел для документа	<section> … </section>
<select>	Представляет элемент управления, который предоставляет меню параметров	<select name=“ ” id=“ ”> <option value=“ ”> … </option> </select>
<small>	Используется для уменьшения шрифта текста на один размер по сравнению с базовым размером	…
<source>	Определяет несколько медиа-ресурсов для разных медиа-элементов, таких как <picture>, <video> и <audio>	<source src=“” type=“audio/mpeg”>
	Используется для стилизации и группировки встроенных элементов	…
<strike>	Используется для отображения зачеркнутого текста (не поддерживается в HTML5)	<strike> … </strike>
	Используется для определения важного текста	…

(продолжение)

<style>	Используется для хранения информации о стиле CSS для HTML-документа	<style> h1{color:blue;} </style>
<sub>	Определяет текст, который отображается в виде нижнего индекса	_…
<Заключение>	Определяет заключение, которое можно использовать с тегом <details>	<Заключение> … </Заключение>
<sup>	Определяет текст, который представляет собой надстрочный текст	[…]
<svg>	Используется как контейнер SVG (масштабируемая векторная графика)	<svg width=“” height=“”> … </svg>

T

<table>	Используется для представления данных в табличной форме или для создания таблицы в документе HTML	<table> … </table>
<tbody>	Представляет основное содержимое HTML-таблицы и используется вместе с <thead> и <tfoot>	<tbody> … </tbody>
<td>	Используется для определения ячеек таблицы HTML, содержащей данные таблицы	<td> … </td>

<template>	Используется для хранения содержимого на стороне клиента, которое не отображается во время загрузки страницы и может отображаться позже с помощью JavaScript	<template>…</template>

(продолжение)

<textarea>	Используется для определения многострочного ввода, например комментария, отзыва, обзора и т.д.	<textarea id="" name="" rows="" cols=""> … </textarea>
<tfoot>	Определяет содержимое нижнего колонтитула HTML-таблицы	<tfoot> … </tfoot>
<th>	Определяет головную ячейку HTML-таблицы	<th> … </th>
<thead>	Определяет заголовок HTML-таблицы. Используется вместе с тегами <tbody> и <tfoot>	<thead> … </thead>
<time>	Используется для определения даты/времени в HTML-документе	<time datetime="2022-02-24 21:00">
<title>	Определяет заголовок или имя HTML-документа	<title> … </title>
<tr>	Определяет ячейки строк в таблице HTML	<tr> … </tr>

U

<u>	Used to render enclosed text with an underline	<u> … </u>
	Определяет неупорядоченный список элементов	…

V

<var>	Определяет имя переменной, используемое в математическом или программном	<var> … </var>
<video>	Используется для встраивания видеоконтента в HTML-документ	<video width="" height="" controls> … </video>

W

<wbr>	Определяет позицию в тексте, где возможен разрыв строки	<wbr> … </wbr>

ПРИЛОЖЕНИЕ В

Справочник по селекторам CSS

A

accent-color	Задает цвет акцента для элементов управления	accent-color: Blue;
gn-content	Задает выравнивание между строками внутри гибкого контейнера, когда элементы не используют все доступное пространство	align-content: center | start | end | normal | stretch | baseline;
align-items	Задает выравнивание элементов внутри гибкого контейнера	align-items: center | start | end | normal | stretch | baseline;
align-self	Задает выравнивание выбранных элементов внутри гибкого контейнера	align-self: center | start | end | normal | stretch | baseline;
all	Сбрасывает все свойства (кроме unicode-bidi и направления	all: initial | inherit | unset;

B

background-color	Определяет цвет фона элемента	background-color: Grey
background-image	Указывает одно или несколько фоновых изображений для элемента	background-image: url("…");
background-origin	Указывает исходное положение фонового изображения	background-origin: padding-box | border-box | content-box | initial | inherit;
background-position	Определяет положение фонового изображения	background-position: left top | left center | left bottom | right top | right center | right bottom | center top | center center | center bottom
background-repeat	Устанавливает, будет ли и как повторяться фоновое изображение	background-repeat: repeat | repeat-x | repeat-y | no-repeat | initial | inherit;
background-size	Определяет размер фоновых изображений	background-size: auto | length | cover | contain | initial | inherit;
border	Сокращенное свойство для ширины границы, стиля границы и цвета границы	border: border-width border-style border-color | initial | inherit;
border-color	Устанавливает цвет четырех границ	border-color: Red;

border-spacing	Устанавливает расстояние между границами соседних ячеек. Используйте длину, чтобы указать размер	border-spacing: length | initial | inherit;

(продолжение)

border-style	Устанавливает стиль четырех границ	border-style: none | hidden | dotted | dashed | solid | double | inherit;
border-width	Устанавливает ширину четырех границ. Используйте длину, чтобы указать размер	border-width: length | initial | inherit | medium | thin | thick;
bottom	Устанавливает положение элемента от нижней части его родительского элемента. Используйте длину, чтобы указать размер	bottom: auto | length | initial | inherit;
box-shadow	Прикрепляет одну или несколько теней к элементу	box-shadow: none | h-offset v-offset blur spread color;

C

@charset	Указывает кодировку символов, используемую в таблице стилей	@charset “UTf-8”;
clear	Указывает, что должно произойти с элементом, находящимся рядом с плавающим элементом	clear: none | left | right | both | initial | inherit;
color	Устанавливает цвет текста	color: Green;
column-count	Указывает количество столбцов, на которые должен быть разделен элемент	column-count: 3;
column-fill	Указывает, как заполнять столбцы, сбалансированные или нет	column-fill: balance | auto | initial | inherit;
column-span	Указывает, сколько столбцов должен охватывать элемент	column-span: none | all | initial | inherit;

column-width	Определяет ширину столбца	
columns	Укороченное свойство для ширины столбца и количества столбцов	column-width: 100px;
cursor	Указывает курсор мыши, который будет отображаться при наведении указателя на элемент	cursor: pointer | help | wait | grab | n-resize;

D

display	Указывает, как должен отображаться определенный элемент HTML	display: inline | block;

F

float	Указывает, должен ли элемент перемещаться влево, вправо или не перемещаться вообще	float: none | left | right | initial | inherit;
font	Краткое свойство для свойств шрифта-стиля, шрифта-варианта, шрифта-веса, размера шрифта/высоты строки и шрифта-семейства	font: 24px Roboto, sans-serif;
font-family	Указывает семейство шрифтов для текста	font-family: Helvetica;
font-kerning	Управляет использованием информации о кернинге (расположение букв)	font-kerning: auto | normal | none;
font-size	Определяет размер шрифта текста. Используйте длину, чтобы указать размер шрифта, например, 12 пикселей	font-size: small | medium | large | length | initial | inherit;

(продолжение)

font-style	Определяет стиль шрифта для текста	font-style: normal | italic | oblique | initial | inherit;
font-weight	Определяет вес шрифта	font-weight: 900;font-weight: bold;

H

height	Устанавливает высоту элемента	height: 20px;

I

image-rendering	Указывает тип алгоритма, который будет использоваться для масштабирования	image-rendering: auto | smooth | high-quality | crisp-edges | pixelated | initial | inherit;
@import	Позволяет импортировать таблицу стилей в другую таблицу стилей	@import url (“…”);@import “…”;

J

justify-content	Определяет выравнивание между элементами внутри гибкого контейнера, когда элементы не используют все доступное пространство.	justify-content: flex-start | flex-end | center;

K

@keyframes	Указывает код анимации	@keyframes move {
		from {top: 0px;}
		to {top: 120px;}
		}

L

left	Указывает левое положение позиционируемого элемента	left: auto | 100px;
letter-spacing	Увеличивает или уменьшает расстояние между символами в тексте	letter-spacing: 2px;
line-break	Указывает, как разрывать строки	line-break: auto | loose | normal | strict | anywhere;
line-height	Устанавливает высоту строки	line-height: length;

M

margin	Устанавливает все свойства полей в одном объявлении	margin: 5px;
margin-bottom	Устанавливает нижнее поле элемента	margin-bottom: 5px;
margin-left	Устанавливает левое поле элемента	margin-left: 5px;
margin-right	Устанавливает правое поле элемента	margin-right: 5px;
margin-top	Устанавливает верхнее поле элемента	margin-top: 5px;

max-height Устанавливает максимальную высоту элемента max-height: 110px;

max-width Устанавливает максимальную ширину элемента max-width: 400px;

@media	Устанавливает правила стиля для различных типов/устройств/размеров носителей	@media only screen and (max-width: 600px) { body { … } }
min-height	Устанавливает минимальную высоту элемента	min-height: 100;
min-width	Устанавливает минимальную ширину элемента	min-width: 600px;

O

object-position	Задает выравнивание заменяемого элемента внутри его поля. Используйте pos, чтобы указать положение	object-position: pos | initial | inherit;object-position: 5px 6px;
opacity	Устанавливает уровень непрозрачности элемента. 0.0 полностью прозрачен; 1.0 полностью непрозрачен	opacity: 0.5;
order	Устанавливает порядок гибкого элемента относительно остальных	order: number | initial | inherit;
outline	Краткое свойство для свойств Outline-width, Outline-style и Outline-color	outline: 10px dotted red;

outline-color	Устанавливает цвет контура	outline-color: red;

outline-offset	Смещает контур и выводит его за пределы границы	outline-offset: 5px;
outline-style	Устанавливает стиль контура	outline-style: none | hidden | dotted | dashed | solid | double | groove | ridge | inset | outset;
outline-width	Устанавливает ширину контура	outline-width: 5px;
overflow	Указывает, что произойдет, если содержимое выйдет за пределы поля элемента	overflow: visible | hidden | clip | scroll | auto | initial | inherit;
overflow-wrap	Указывает, может ли браузер разбивать строки с длинными словами, если они выходят за пределы контейнера	overflow-wrap: normal | anywhere | break-word | initial | inherit;

P

padding	Сокращенное свойство для всех свойств заполнения	padding: 10px;
padding-bottom	Устанавливает нижний отступ элемента	padding-bottom: 10px;
padding-left	Устанавливает отступы слева от элемента	padding-left: 10px;
padding-right	Устанавливает правильное отступы элемента	padding-right: 10px;
padding-top	Устанавливает верхнее отступ элемента	padding-top: 10px;

position	Указывает тип метода позиционирования, используемого для элемента	position: static | absolute | fixed | relative | sticky | initial | inherit;

R

resize	Определяет, как пользователь может изменить размер элемента	resize: none | both | horizontal | vertical | initial | inherit;
right	Указывает правильное положение позиционированного элемента	right: 10px;
row-gap	Определяет разрыв между строками сетки	row-gap: 5px;

S

scroll-behavior	Указывает, следует ли плавно анимировать положение прокрутки в прокручиваемом поле вместо прямого перехода	scroll-behavior: auto | smooth | initial | inherit;

T

tab-size	Определяет ширину символа табуляции	tab-size: 10;
table-layout	Определяет алгоритм, используемый для расположения ячеек, строк и столбцов таблицы	table-layout: auto | fixed | initial | inherit;
text-align	Задает горизонтальное выравнивание текста	text-align: left | right | center | justify | initial | inherit;
text-decoration	Определяет украшение, добавляемое к тексту	text-decoration: overline | line-through | underline;

text-decoration-color	Определяет цвет текстового оформления	text-decoration-color: red;
text-indent	Определяет отступ первой строки в текстовом блоке	text-indent: 40px;
text-shadow	Добавляет тень к тексту	text-shadow: 2px 2px lightgrey;
top	Указывает верхнюю позицию позиционируемого элемента	top: 5px;

U

user-select	Указывает, можно ли выбрать текст элемента	user-select: auto | none | text | all;

V

vertical-align	Устанавливает вертикальное выравнивание элемента	vertical-align: baseline | length | sub | super | top | text-top | middle | bottom | text-bottom | initial | inherit;
visibility	Указывает, виден ли элемент	visibility: visible | hidden | collapse | initial | inherit;

W

width	Устанавливает ширину elementwidth: 150 пикселей;	width: auto | value | initial | inherit;
word-spacing	Увеличивает или уменьшает расстояние между словами в тексте	word-spacing: 3px;
writing-mode	Указывает, располагаются ли строки текста горизонтально или вертикально	writing-mode: vertical-rl;

ПРИЛОЖЕНИЕ C

Коды цветов CSS

Вы можете указать цвета, используя следующие форматы:

- Ключевое слово, например “red”, “green”, “blue”, “transparent”, “orange” и т.д.
- Шестнадцатеричное значение, например “#000000”, “#00A500”, “#FFFFFF” и т.д.
- Значение RGB, например “rgb(255, 255, 0)”

Ключевое слово	Шестнадцатеричное	RGB-значение
aliceblue	#f0f8ff	rgb(240, 248, 255)
antiquewhite	#fAEBD7	rgb(250, 235, 215)
aqua	#00ffff	rgb(0, 255, 255)
aquamarine	#7fffD4	rgb(127, 255, 212)
azure	#f0ffff	rgb(1240, 255, 255)
beige	#f5f5DC	rgb(245, 245, 220)
bisque	#ffE4C4	rgb(255, 228, 196)
black	#000000	rgb(0, 0, 0)
blanchedalmond	#ffEBCD	rgb(255, 235, 205)
blue	#0000ff	rgb(0, 0, 255)

(продолжение)

Ключевое слово	Шестнадцатеричное	RGB-значение
blueviolet	#8A2BE2	rgb(138, 43, 226)
brown	#A52A2A	rgb(165, 42, 42)
burlywood	#DEB887	rgb(222, 184, 135)
cadetblue	#5f9EA0	rgb(95, 158, 160)
chartreuse	#7fff00	rgb(95, 158, 160)
chocolate	#D2691E	rgb(210, 105, 30)
coral	#ff7f50	rgb(255, 127, 80)
cornflowerblue	#6495ED	rgb(100, 149, 237)
cornsilk	#fff8DC	rgb(255, 248, 220)
crimson	#DC143C	rgb(220, 20, 60)
cyan	#00ffff	rgb(0, 255, 255)
darkblue	#00008B	rgb(0, 0, 139)
darkcyan	#008B8B	rgb(0, 139, 139)
darkgoldenrod	#B8860B	rgb(184, 134, 11)
darkgray	#A9A9A9	rgb(169, 169, 169)
darkgreen	#006400	rgb(0, 100, 0)
darkkhaki	#BDB76B	rgb(189, 183, 107)
darkmagenta	#8B008B	rgb(139, 0, 139)
darkolivegreen	#556B2f	rgb(85, 107, 47)
darkorange	#ff8C00	rgb(255, 140, 0)
darkorchid	#9932CC	rgb(153, 50, 204)
darkred	#8B0000	rgb(139, 0, 0)
darksalmon	#E9967A	rgb(233, 150, 122)

(продолжение)

Ключевое слово	Шестнадцатеричное	RGB-значение
darkseagreen	#8fBC8f	rgb(143, 188, 143)
darkslateblue	#483D8B	rgb(72, 61, 139)
darkslategray	#2f4f4f	rgb(47, 79, 79)
darkturquoise	#00CED1	rgb(0, 206, 209)
darkviolet	#9400D3	rgb(148, 0, 211)
deeppink	#ff1493	rgb(255, 20, 147)
deepskyblue	#00Bfff	rgb(0, 191, 255)
dimgray	#696969	rgb(0, 191, 255)
dodgerblue	#1E90ff	rgb(30, 144, 255)
firebrick	#B22222	rgb(178, 34, 34)
floralwhite	#fffAf0	rgb(255, 250, 240)
forestgreen	#228B22	rgb(34, 139, 34)
fuchsia	#ff00ff	rgb(255, 0, 255)
gainsboro	#DCDCDC	rgb(220, 220, 220)
ghostwhite	#f8f8ff	rgb(248, 248, 255)
gold	#ffD700	rgb(255, 215, 0)
goldenrod	#DAA520	rgb(218, 165, 32)
gray	#808080	rgb(128, 128, 128)
green	#008000	rgb(0, 128, 0)
greenyellow	#ADff2f	rgb(173, 255, 47)
honeydew	#f0fff0	rgb(240, 255, 240)
hotpink	#ff69B4	rgb(255, 105, 180)
indianred	#CD5C5C	rgb(205, 92, 92)

(*продолжение*)

Ключевое слово	Шестнадцатеричное	RGB-значение
indigo	#4B0082	rgb(75, 0, 130)
ivory	#fffff0	rgb(255, 255, 240)
khaki	#f0E68C	rgb(240, 230, 140)
lavender	#E6E6fA	rgb(230, 230, 250)
lavenderblush	#fff0f5	rgb(255, 240, 245)
lawngreen	#7CfC00	rgb(124, 252, 0)
lemonchiffon	#fffACD	rgb(255, 250, 205)
lightblue	#ADD8E6	rgb(173, 216, 230)
lightcoral	#f08080	rgb(240, 128, 128)
lightcyan	#E0ffff	rgb(224, 255, 255)
lightgoldenrodyellow	#fAfAD2	rgb(250, 250, 210)
lightgreen	#90EE90	rgb(144, 238, 144)
lightgrey	#D3D3D3	rgb(211, 211, 211)
lightpink	#ffB6C1	rgb(255, 182, 193)
lightsalmon	#ffA07A	rgb(255, 160, 122)
lightseagreen	#20B2AA	rgb(32, 178, 170)
lightskyblue	#87CEfA	rgb(135, 206, 250)
lightslategray	#778899	rgb(119, 136, 153)
lightsteelblue	#B0C4DE	rgb(176, 196, 222)
lightyellow	#ffffE0	rgb(255, 255, 224)
lime	#00ff00	rgb(0, 255, 0)
limegreen	#32CD32	rgb(50, 205, 50)
linen	#fAf0E6	rgb(250, 240, 230)

(продолжение)

Ключевое слово	Шестнадцатеричн	RGB-значение
magenta	#ff00ff	rgb(255, 0, 255)
maroon	#800000	rgb(128, 0, 0)
mediumaquamarine	#66CDAA	rgb(102, 205, 170)
mediumblue	#0000CD	rgb(0, 0, 205)
mediumorchid	#BA55D3	rgb(186, 85, 211)
mediumpurple	#9370DB	rgb(147, 112, 219)
mediumseagreen	#3CB371	rgb(60, 179, 113)
mediumslateblue	#7B68EE	rgb(123, 104, 238)
mediumspringgreen	#00fA9A	rgb(0, 250, 154)
mediumturquoise	#48D1CC	rgb(72, 209, 204)
mediumvioletred	#C71585	rgb(199, 21, 133)
midnightblue	#191970	rgb(25, 25, 112)
mintcream	#f5fffA	rgb(245, 255, 250)
mistyrose	#ffE4E1	rgb(255, 228, 225)
moccasin	#ffE4B5	rgb(255, 228, 181)
navajowhite	#ffDEAD	rgb(255, 222, 173)
navy	#000080	rgb(0, 0, 128)
navyblue	#9fAfDf	rgb(159, 175, 223)
oldlace	#fDf5E6	rgb(253, 245, 230)
olive	#808000	rgb(128, 128, 0)
olivedrab	#6B8E23	rgb(107, 142, 35)
orange	#ffA500	rgb(255, 165, 0)
orangered	#ff4500	rgb(255, 69, 0)

(*продолжение*)

Ключевое слово	Шестнадцатеричное	RGB-значение
orchid	#DA70D6	rgb(218, 112, 214)
palegoldenrod	#EEE8AA	rgb(238, 232, 170)
palegreen	#98fB98	rgb(152, 251, 152)
paleturquoise	#AfEEEE	rgb(175, 238, 238)
palevioletred	#DB7093	rgb(219, 112, 147)
papayawhip	#ffEfD5	rgb(255, 239, 213)
peachpuff	#ffDAB9	rgb(255, 218, 185)
peru	#CD853f	rgb(205, 133, 63)
pink	#ffC0CB	rgb(255, 192, 203)
plum	#DDA0DD	rgb(221, 160, 221)
powderblue	#B0E0E6	rgb(176, 224, 230)
purple	#800080	rgb(128, 0, 128)
red	#ff0000	rgb(255, 0, 0)
rosybrown	#BC8f8f	rgb(188, 143, 143)
royalblue	#4169E1	rgb(65, 105, 225)
saddlebrown	#8B4513	rgb(139, 69, 19)
salmon	#fA8072	rgb(250, 128, 114)
sandybrown	#fA8072	rgb(244, 164, 96)
seagreen	#2E8B57	rgb(46, 139, 87)
seashell	#fff5EE	rgb(255, 245, 238)
sienna	#A0522D	rgb(160, 82, 45)
silver	#C0C0C0	rgb(192, 192, 192)
skyblue	#87CEEB	rgb(135, 206, 235)

(продолжение)

Ключевое слово	Шестнадцатеричное	RGB-значение
slateblue	#6A5ACD	rgb(106, 90, 205)
slategray	#708090	rgb(112, 128, 144)
snow	#fffAfA	rgb(255, 250, 250)
springgreen	#00ff7f	rgb(0, 255, 127)
steelblue	#4682B4	rgb(70, 130, 180)
tan	#D2B48C	rgb(210, 180, 140)
teal	#008080	rgb(0, 128, 128)
thistle	#D8BfD8	rgb(216, 191, 216)
tomato	#ff6347	rgb(255, 99, 71)
turquoise	#40E0D0	rgb(64, 224, 208)
violet	#EE82EE	rgb(238, 130, 238)
wheat	#f5DEB3	rgb(245, 222, 179)
white	#ffffff	rgb(255, 255, 255)
whitesmoke	#f5f5f5	rgb(245, 245, 245)
yellow	#ffff00	rgb(255, 255, 0)
yellowgreen	#9ACD32	rgb(139, 205, 50)