Структура документа и веб-сайта

В дополнение к определению отдельных частей вашей страницы (таких как «абзац» или «изображение»), HTML также содержит ряд элементов блочного уровня, используемых для определения областей вашего веб-сайта (такие как «заголовок», «навигационное меню», «колонка основного содержимого»). В этой статье рассматривается, как планировать базовую структуру сайта и писать HTML для представления этой структуры.

Необходимые знания:	**Базовое знакомство с HTML, описано в разделе Начало работы с HTML. Форматирование текста в HTML, описано в разделе Основы текста в HTML. Как работают гиперссылки, описано в разделе Создание гиперссылок.**
Задача:	Изучить, как структурировать документ с помощью семантических тегов и как разработать структуру простого веб-сайта.

Основные составляющие документа

Веб-страницы могут и будут отличаться друг от друга, но все они, преимущественно, состоят из аналогичных стандартных компонентов, если только страница не отображает полноэкранное видео или игру, не является частью какого-либо художественного проекта или просто плохо структурирована:

Заголовок (колонтитул)

Обычно это большая полоса вверху страницы, с крупным заголовком и / или логотипом. Здесь указывается общая информация о веб-сайте, не меняющаяся от страницы к странице.

Навигационное меню

Ссылки на основные разделы сайта; обычно в виде кнопок, ссылок или вкладок. Также как и заголовок, навигация остаётся неизменной на всех страницах сайта — наличие непоследовательной навигации на вашем сайте запутает и разочарует пользователей. Многие веб-дизайнеры считают панель навигации частью заголовка, а не отдельным компонентом, но это не является обязательным требованием; на самом деле, некоторые также утверждают, что их разделение на отдельные компоненты улучшает доступность, поскольку раздельная структура будет понятнее для людей, пользующихся считывателями экрана.

Основное содержимое

Большая область в центре страницы, содержащая, в основном, уникальный контент данной веб-страницы, например видео, которое вы хотите посмотреть, или рассказ, который вы читаете, или карту, которую вы хотите просмотреть, или заголовки новостей и т. д. Это одна из частей сайта, которая определённо будет меняться от страницы к странице!

Боковая панель

Как правило, содержит некоторую второстепенную информацию, ссылки, цитаты, рекламу и т.д. Обычно она относится к содержимому в основном контенте (например, на странице со статьёй, боковая панель может содержать биографию автора или ссылки на связанные статьи), но в некоторых случаях здесь размещают и другие элементы, например, вторичную навигационную систему.

Нижний колонтитул (футер)

Полоса в нижней части страницы, которая обычно содержит уведомления об авторских правах или контактную информацию. Это место для размещения общей информации (например, заголовка), но обычно эта информация не является критичной или вторична для самого веб-сайта. Нижний колонтитул также иногда используется для SEO целей, предоставляя ссылки для быстрого доступа к популярному контенту.

«Типичный веб-сайт» может быть структурирован примерно так:

HTML для структурирования содержимого

Пример, показанный сверху, не красив и примитивен, но идеально подходит для иллюстрирования типичного макета веб-сайта. У некоторых веб-сайтов больше колонок, некоторые — более сложные, но идею вы поняли. С правильным CSS вы могли бы использовать практически любые элементы для обёртывания различных разделов и стилизовать их так, как вам хочется, но, как обсуждалось ранее, нам нужно уважать семантику и **использовать правильный элемент для правильной работы.**

Это потому, что визуальные эффекты — это ещё не самое главное. Мы используем цвет и размер шрифта для привлечения внимания посетителей к наиболее полезным частям содержимого, такого как навигационное меню или связанные ссылки, но что насчёт людей со

слабым зрением, к примеру, для которых концепция «розового» и «большого шрифта» не будет полезной?

Примечание: Люди с дальтонизмом составляют около 8% мирового населения. Слепые и слабовидящие люди составляют примерно 4-5% населения мира (в 2012 году в мире было 285 миллионов таких людей, а общая численность населения составляла около 7 миллиардов).

В своём HTML-коде вы можете размечать разделы содержимого сайта на основе их функциональности — использовать элементы, которые представляют разделы контента, описанные выше, а вспомогательные технологии, такие как программы чтения с экрана, смогут распознавать эти элементы и помогать в таких задачах, как «найти основную навигацию» или «найти основное содержимое». Как мы упоминали ранее в ходе курса, существует ряд последствий неиспользования правильной структуры элементов и семантики для правильной работы.

Для реализации такой семантической разметки HTML предоставляет выделенные теги, которые можно использовать для создания таких разделов, например:

- Заголовок: <header>.
- Навигационное меню: <nav>.
- Основное содержимое: <main>, с различными подразделами содержимого, представленными элементами <article>, <section> и .
- Боковая панель: <aside>, обычно располагается внутри <main>.
- Нижний колонтитул: <footer>.

Активное обучение: исследование кода для нашего примера

Наш пример, представленный выше, содержит следующий код (Вы также можете найти пример в нашем репозитории Github). Мы хотели бы, чтобы вы взглянули на приведённый выше пример, а затем просмотрели код ниже, чтобы узнать, из каких частей он состоит.

HTML

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8" />

    <title>Заголовок моей страницы</title>
    <link
href="https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300|Sonsie+One"
      rel="stylesheet"
```

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

    <!-- следующие 3 строки нужны для корректного отображения семантических
элементов HTML5 в старых версиях Internet Explorer-->
    <!--[if lt IE 9]>
      <script
src="https://cdnjs.cloudflare.com/ajax/libs/html5shiv/3.7.3/html5shiv.j
s"></script>
    <![endif]-->
  </head>

  <body>
    <!-- Вот наш главный заголовок, который используется на всех страницах нашего
веб-сайта -->

    <header>
      <h1>Заголовок</h1>
    </header>

    <nav>
      <ul>
        <li><a href="#">Домашняя страница</a></li>
        <li><a href="#">Наша команда</a></li>
        <li><a href="#">Проекты</a></li>
        <li><a href="#">Контакты</a></li>
      </ul>

      <!-- Форма поиска — это ещё один распространённый нелинейный способ
навигации по веб-сайту. -->

      <form>
        <input type="search" name="q" placeholder="Search query" />
        <input type="submit" value="Go!" />
      </form>
    </nav>

    <!-- Здесь основное содержимое нашей страницы -->
    <main>
      <!-- Она содержит статью -->
      <article>
        <h2>Заголовок статьи</h2>

        <p>
          Lorem ipsum dolor sit amet, consectetur adipisicing elit.
Donec a diam
          lectus. Set sit amet ipsum mauris. Maecenas congue ligula as
quam
          viverra nec consectetur ant hendrerit. Donec et mollis dolor.
Praesent
          et diam eget libero egestas mattis sit amet vitae augue. Nam
```

```
tincidunt
          congue enim, ut porta lorem lacinia consectetur.
        </p>

        <h3>Подраздел</h3>

        <p>
          Donec ut librero sed accu vehicula ultricies a non tortor.
Lorem ipsum
          dolor sit amet, consectetur adipisicing elit. Aenean ut
gravida lorem.
          Ut turpis felis, pulvinar a semper sed, adipiscing id dolor.
        </p>

        <p>
          Pelientesque auctor nisi id magna consequat sagittis.
Curabitur
          dapibus, enim sit amet elit pharetra tincidunt feugiat nist
imperdiet.
          Ut convallis libero in urna ultrices accumsan. Donec sed odio
eros.
        </p>

        <h3>Ещё один подраздел</h3>

        <p>
          Donec viverra mi quis quam pulvinar at malesuada arcu
rhoncus. Cum
          soclis natoque penatibus et manis dis parturient montes,
nascetur
          ridiculus mus. In rutrum accumsan ultricies. Mauris vitae
nisi at sem
          facilisis semper ac in est.
        </p>

        <p>
          Vivamus fermentum semper porta. Nunc diam velit, adipscing ut
          tristique vitae sagittis vel odio. Maecenas convallis
ullamcorper
          ultricied. Curabitur ornare, ligula semper consectetur
sagittis, nisi
          diam iaculis velit, is fringille sem nunc vet mi.
        </p>
      </article>

      <!-- Дополнительный контент также может быть вложен в основной контент -->
      <aside>
        <h2>Связанные темы</h2>

        <ul>
          <li><a href="#">Мне нравится стоять рядом с берегом моря</a></li>
```

```
            <li><a href="#">>Мне нравится стоять рядом с морем</a></li>
            <li><a href="#">Даже на севере Англии</a></li>
            <li><a href="#">Здесь не перестаёт дождь</a></li>
            <li><a href="#">Лаааадно...</a></li>
          </ul>
        </aside>
      </main>

      <!-- И вот наш главный нижний колонтитул, который используется на всех
страницах нашего веб-сайта -->

      <footer>
        <p>©Авторские права никому не принадлежат, 2050. Все права защищены.</p>
      </footer>
    </body>
</html>
```

Потратьте некоторое время, чтобы просмотреть код и понять его — комментарии внутри кода также помогут вам в этом. Мы не просим вас делать ничего больше в этом уроке, потому что ключ к пониманию макета документа заключается в написании осмысленной структуры HTML, а затем её развёртывании с помощью CSS. Мы подождём, пока вы не начнёте изучать CSS-макет как часть темы CSS.

Подробнее об элементах HTML макета

Полезно понять общий смысл всех структурных элементов HTML — это то, над чем вы будете работать постепенно, когда начнёте получать больше опыта с веб-разработкой. Вы можете ознакомиться с деталями, прочитав статью HTML-элементы. Пока что это основные определения, которые вы должны попытаться понять:

- <main> предназначен для содержимого, уникального для этой страницы. Используйте <main> только один раз на странице и размещайте прямо внутри <body>. В идеале он не должен быть вложен в другие элементы.
- <article> окружает блок связанного содержимого, который имеет смысл сам по себе без остальной части страницы (например, один пост в блоге).
- <section> подобен <article>, но больше подходит для группирования одной части страницы, которая представляет собой одну часть функциональности (например, мини-карту или набор заголовков статей и сводок). Считается хорошей практикой начинать каждый раздел с заголовка. Также обратите внимание, что в зависимости от контекста вы можете разбить <article> на несколько <section> или, наоборот, <section> на несколько <article>.
- <aside> содержит контент, который не имеет прямого отношения к основному содержимому, но может содержать дополнительную информацию, косвенно связанную с ним (словарь, биография автора, связанные ссылки и т. д.).
- <header> представляет собой группу вводного содержимого. Если он дочерний элемент <body>, то он определяет глобальный заголовок веб-страницы, но если он дочерний элемент <article> или <section>, то определяет конкретный заголовок для этого раздела (постарайтесь не путать его с titles и headings).

- <nav> содержит основные функции навигации для страницы. Так же часто в нем можно увидеть логотип и / или название сайта или компании. Вторичные ссылки и т. д. не входят в навигацию.
- <footer> представляет собой группу конечного контента для страницы.

Несемантические обёртки

Иногда вы будете сталкиваться с ситуацией, когда вы не можете найти идеальный семантический элемент, чтобы сгруппировать некоторые элементы вместе или обернуть некоторый контент. Иногда вам просто нужно будет сгруппировать несколько элементов вместе, чтобы применить к ним, как к единой сущности, CSS или JavaScript. Для таких случаев в HTML есть элементы и . Вам следует использовать их с подходящим значением атрибута class или id, чтобы можно было легко получить к ним доступ.

 — это строчный несемантический элемент, который стоит использовать только если вы не можете подобрать более подходящий семантический текстовый элемент для обёртывания контента или если не хотите добавлять какие-либо конкретные значения. Например:

HTML

```
<p>
 Пьяный Король возвратился в свою комнату в 01:00 и всё никак не мог войти в
 дверь: хмель мешал
  <span class="editor-note"
    >[Примечание редактора: В этот момент свет на сцене должен быть
  приглушён]</span
  >.
</p>
```

В этом примере примечание редактора просто сообщает дополнительные пожелания режиссёру пьесы. В нем нет особого семантического значения. Для слабовидящих пользователей, возможно, примечание будет отделено от основного содержимого с помощью CSS.

 — это блочный несемантический элемент, который следует использовать только если вы не можете подобрать более подходящий семантический блочный элемент или если не хотите добавлять какие-либо конкретные значения. Например, представьте виджет корзины в интернет-магазине, который вы можете открыть в любой момент нахождения на сайте:

HTML

```
<div class="shopping-cart">
  <h2>Корзина</h2>
  <ul>
    <li>
      <p>
        <a href=""><strong>Silver earrings</strong></a
        >: $99.95.
```

```
      </p>
      <img src="../products/3333-0985/thumb.png" alt="Серебряные серьги"
/>
    </li>
    <li>...</li>
  </ul>
  <p>Итого: $237.89</p>
</div>
```

Ему не подходит <aside>, поскольку это не обязательно относится к основному содержимому страницы (Вы хотите, чтобы его можно было просматривать из любого места). Также не подходит и <section>, т. к. это не часть основного содержимого страницы. Поэтому подходит в этом случае. Мы включили заголовок в качестве указателя, чтобы помочь пользователям программ чтения с экрана в его поиске.

Предупреждение: Внимание: div настолько просто использовать, что легко переборщить. Поскольку они не несут никакого семантического значения, они просто загромождают ваш HTML-код. Старайтесь использовать их только тогда, когда нет лучшего семантического решения, и постарайтесь свести их использование к минимуму, иначе вам будет трудно обновлять и поддерживать ваши документы.

Перенос строки и горизонтальный разделитель

Два элемента, которые вы будете периодически использовать или захотите узнать о них:
 и <hr>:

 создаёт разрыв строки в абзаце, и это единственный способ изменить жёсткую структуру в ситуации, когда вам нужна серия фиксированных коротких строк, например, в почтовом адресе или стихотворении. Пример:

HTML

```
<p>
 Жила-была девчушка Нелл,<br />
 Любившая писать  HTML:<br />
 Её семантика ужасна была — <br />
 Она и сама прочитать ничего не могла.
</p>
```

Без элемента
 абзац разместится в одну длинную линию (как было сказано ранее, HTML игнорирует переносы строк), а с ним в коде — разметка будет выглядеть следующим образом:

Жила-была девчушка Нелл, Любившая писать HTML: Её семантика ужасна была — Она и сама

прочитать ничего не могла.

<hr> создаёт горизонтальный разделитель в документе, это означает тематическое изменение текста (например, изменение темы или сцены). Визуально он просто похож на горизонтальную линию. В качестве примера:

HTML

```
<p>Рон был зажат в углу адскими тварями. Он боялся, но твёрдо решил защитить своих друзей, поднял свою волшебную палочку и приготовился к битве, надеясь, что справится со своим несчастьем.</p>
<hr>
<p>Тем временем Гарри сидел дома с раскрытым указом и размышлял о том, когда выйдут новые серии спин-оффов; в это время зачарованное письмо пархнуло в окно и приземлилось у него на коленях. Он прочитал его и подскочил на ноги; он подумал: "Думаю, самое время вернуться к работе".</p>
```

Будет выглядеть примерно так:

Создание первого HTML-элемента

Отредактируйте строку ниже в области «Редактора кода», обернув ее тегами и .Чтобы открыть элемент, поместите открывающий тег в начало строки. Чтобы закрыть элемент, поместите закрывающий тег в конце строки. Это должно привести к форматированию текста *курсивом*! Просматривайте обновления изменений в режиме реального времени в области «Вывод (правая область редактора). Если вы допустили ошибку, вы можете очистить свою работу с помощью кнопки «Reset».

Вложенные элементы

Элементы можно размещать внутри других элементов. Это называется **вложением**. Если бы мы хотели указать, что наш **кот** очень сварливый, мы могли бы обернуть слово «очень» в элемент, что означает, что слово должно иметь строгое форматирование текста:

HTML

```
<p>Мой кот <strong>очень</strong> сварливый.</p>
```

Пример:

Мой кот **очень** сварливый.

Есть правильный и неправильный способ вложения. В приведенном выше примере мы сначала открыли элемент тегом ** **, а затем открыли тег ****. Для правильной вложенности мы должны сначала закрыть элемент тегом ****, а затем закрыть весь элемент ** **.

Ниже приведен пример неправильного способа вложения:

HTML

```
<p>Мой кот <strong>очень сварливый.</p></strong>
```

Пример:

Мой кот **очень сварливый.**

Теги должны открываться и закрываться таким образом, чтобы они находились внутри или снаружи друг друга. Учитывая такое перекрытие, как в приведенном выше примере, браузеру приходится угадывать ваши намерения. Подобные догадки могут привести к неожиданным результатам.

Пустотные элементы

Не все элементы следуют шаблону открывающего тега, содержимого и закрывающего тега. Некоторые элементы состоят из одного тега, который обычно используется для вставки/внедрения чего-либо в документ. Такие элементы называются пустыми элементами. Например, <img> элемент встраивает файл изображения на страницу:

HTML

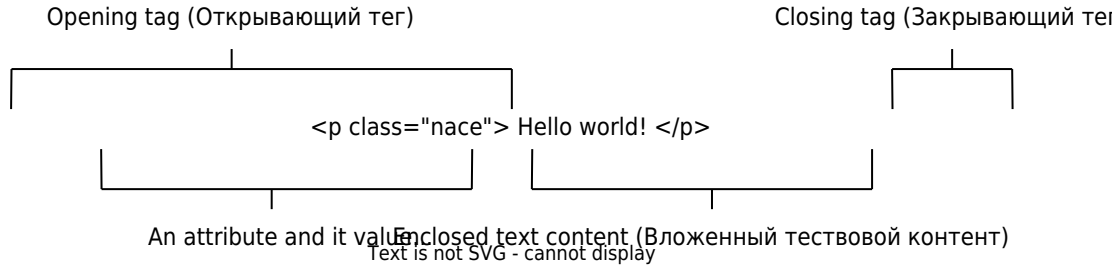
```
<img
  src="images/firefox-icon.png"
  alt="Firefox icon" />
```

Это выведет следующее:

Примечание. В HTML нет необходимости добавлять символ / в конце тега элемента void (пустого элемента), например: . Однако это также допустимый синтаксис, и вы можете сделать это, если хотите, чтобы ваш HTML был допустимым XML.

Атрибуты

Элементы также могут иметь атрибуты. Атрибуты выглядят следующим образом:



Атрибуты содержат дополнительную информацию об элементе, которая не будет отображаться в содержимом. В этом примере 'class' атрибут представляет собой идентифицирующее имя, используемое для указания элемента с информацией о стиле. Атрибут должен иметь:

- Пробел между ним и именем элемента. (Для элемента с более чем одним атрибутом атрибуты также должны быть разделены пробелами.)
- Имя атрибута, за которым следует знак равенства.
- Значение атрибута, заключенное в открывающие и закрывающие кавычки.

Добавление атрибутов к элементу

Другой пример элемента: <a>. Это означает якорь . Якорь может превратить заключенный в него текст в гиперссылку. Якоря могут принимать ряд атрибутов, вот некоторые из них:

href	Значение этого атрибута определяет веб-адрес ссылки. Например: `href=«https://www.mozilla.org/»`.
title	Атрибут `title` указывает дополнительную информацию о ссылке, например описание страницы, на которую имеется ссылка. Например, `title=«The Mozilla homepage»`. Оно появляется в виде всплывающей подсказки при наведении курсора на элемент.
target	Атрибут `target` определяет контекст просмотра, используемый для отображения ссылки. Например, `target=«_blank»` отобразит ссылку в новой вкладке. Если вы хотите отображать связанный контент на текущей вкладке, просто опустите этот атрибут.

Отредактируйте строку ниже в области ввода , чтобы превратить ее в ссылку на ваш любимый веб-сайт.

1. Добавьте открывающий <a> и закрывающий теги в элемент.
2. Добавьте `href` атрибут и через знак = в кавычках ссылку на любой сайт и атрибут `title` с присвоением через знак = информации о сайте заключенной в кавычки.
3. Укажите `target` атрибут, чтобы ссылка открывалась в новой вкладке `(target=«_blank»)`.

Отредактируйте строку ниже в области «Редактора кода», и вы должны увидеть ссылку, которая при наведении курсора мыши отображает значение атрибута `title`, а при нажатии открывает новую вкладку и переходит к веб-адресу `href` атрибута. Помните, что вам необходимо включать пробел между именем элемента и между каждым атрибутом. Просматривайте обновления изменений в режиме реального времени в области «Вывод (правая область редактора).

Если вы допустили ошибку, вы можете очистить свою работу с помощью кнопки «Reset». Смотрите пример ниже:

HTML

```
<a href="https://www.mozilla.org/" title="The Mozilla homepage"
target="_blank">Ссылка на сайт</a>
```

Логические атрибуты

Иногда вы увидите атрибуты, написанные без значений. Это вполне приемлемо. Они называются логическими атрибутами. Логические атрибуты могут иметь только одно значение, которое обычно совпадает с именем атрибута. Например, рассмотрим disabled атрибут, который можно назначить элементам ввода формы. (Это используется для отключения элементов ввода формы, чтобы пользователь не мог вводить данные. Отключенные элементы обычно отображаются серым цветом.) Например:

HTML

```
<!-- использование отключенного атрибута не позволяет конечному пользователю
вводить текст в поле ввода -->
<input type="text" disabled="disabled" />
```

Для краткости допустимо записать это следующим образом:

HTML

```
<!-- использование отключенного атрибута не позволяет конечному пользователю
вводить текст в поле ввода -->
<input type="text" disabled />


<!-- ввод текста разрешен, так как он не содержит атрибута отключен -->
<input type="text" />
```

Для справки: приведенный выше пример также включает неотключенный элемент ввода формы. HTML из приведенного выше примера дает такой результат:

Пропуск кавычек вокруг значений атрибутов

Если вы посмотрите на код многих других сайтов, вы можете встретить ряд странных стилей разметки, включая значения атрибутов без кавычек. Это разрешено при определенных обстоятельствах, но также может привести к нарушению вашей разметки при других обстоятельствах. Например, если мы вернемся к нашему предыдущему примеру со ссылкой, мы могли бы написать базовую версию только с атрибутом `href`, вот так:

HTML

```
<!-- правильный синтаксис (с кавычками) -->
<a href="https://www.mozilla.org/">Ссылка на сайт</a>

<!-- не желательный синтаксис без кавычек (разрешено при определенных обстоятельствах) -->
<a href=https://www.mozilla.org/>Ссылка на сайт</a>
```

Однако как только мы добавляем атрибут `title`таким образом, возникают проблемы:

HTML

```
<!-- не правильный синтаксис (без кавычек) при атрибуте title со значением The (вместо The Mozilla homepage) -->
<a href=https://www.mozilla.org/ title=The Mozilla homepage>Ссылка на сайт</a>
```

Как написано выше, браузер неправильно интерпретирует разметку, принимая `title` атрибут за три атрибута: атрибут `title` со значением The и два логических атрибута `Mozilla` и `homepage`. Очевидно, это не предназначено! Это приведет к ошибкам или неожиданному поведению, как вы можете видеть на живом примере ниже. Попробуйте навести курсор на ссылку, чтобы просмотреть текст заголовка!

Всегда включайте кавычки атрибутов. Это позволяет избежать таких проблем и приводит к более читабельному коду.

Одинарные или двойные кавычки?

В этой статье вы также заметите, что атрибуты заключены в двойные кавычки. Однако в некотором HTML-коде вы можете увидеть одинарные кавычки. Это вопрос стиля. Вы можете смело выбирать, какой из них вам больше по душе. Обе эти строки эквивалентны:

HTML

```
<!-- правильный синтаксис (с двойными кавычками) -->
<a href="https://www.mozilla.org/" title="The Mozilla homepage">Ссылка на сайт </a>

<!-- правильный синтаксис (с одинарными кавычками) -->
<a href='https://www.mozilla.org/' title='The Mozilla homepage'>Ссылка на сайт#1 </a>

<!-- правильный синтаксис (с двойными для атрибута href="" и и одиночными для атрибута title='') -->
<a href="https://www.mozilla.org/" title='The Mozilla homepage'>Ссылка
```

```
на сайт#2 </a>
```

Убедитесь, что вы не смешиваете одинарные и двойные кавычки для одного атрибута. Этот пример (ниже) показывает своего рода смешение кавычек, которое может пойти не так (вместо ссылки https://www.mozilla.org/ получим <color #ed1c24>https://www.mozilla.org/'title=</color>):

HTML

```
<!-- не правильный синтаксис (с разными кавычками) -->
<a href="https://www.mozilla.org/' title="The Mozilla homepage">Ссылка на сайт#3 </a>
```

Однако если вы используете один тип кавычек, вы можете включить кавычки другого типа в значения атрибутов:

HTML

```
<!-- одинарная кавычка внутри двойной -->
<a href="https://www.mozilla.org/" title="Isn't no the Mozilla homepage">Ссылка на сайт#4 </a>
```

Чтобы использовать кавычки внутри других кавычек того же типа (одинарной или двойной кавычки), используйте сущности HTML (например `&quot;` может быть интерпретирован как обрамляющая значение атрибута кавычка):

HTML

```
<!-- использование сущности html &quot;  -->
<a href="https://www.mozilla.org/" title="Isn't &quot;no the&quot; Mozilla homepage">Ссылка на сайт#5 </a>
```

Пример неправильного использования кавычек внутри других кавычек значения атрибута (отредактируйте строку ниже в области ввода в окне редактора **<color #ed1c24>«</color><color #22b14c>no the</color><color #ed1c24>»</color>** на **<color #ed1c24>"</color><color #22b14c>no the</color><color #ed1c24>"</color>** и вы сможете увидеть свои изменения в режиме реального времени в области «Вывод»). Если вы допустили ошибку, вы всегда можете сбросить ее с помощью кнопки Reset:

HTML

```
<!-- использование сущности html &quot;  -->
<a href="https://www.mozilla.org/" title="Isn't "no the" Mozilla homepage">Ссылка на сайт#6 </a>
```

Анатомия HTML-документа

Отдельные элементы HTML сами по себе бесполезны. Далее давайте рассмотрим, как отдельные элементы объединяются, образуя целую HTML-страницу:

HTML

```
<!doctype html>
<html lang="en-US">
  <head>
    <meta charset="utf-8" />
    <title>My test page</title>
  </head>
  <body>
    <p>This is my page</p>
  </body>
</html>
```

Здесь у нас есть:

- <!DOCTYPE html>: Тип документа. Когда HTML был молод (1991-1992), типы документов должны были действовать как ссылки на набор правил, которым должна была следовать HTML-страница, чтобы считаться хорошим HTML. Раньше типы документов выглядели примерно так:

HTML

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

В последнее время тип документа стал историческим артефактом, который необходимо включить, чтобы все остальное работало правильно. <!DOCTYPE html>— это кратчайшая строка символов, которая считается допустимым типом документа. Это все, что вам нужно знать!

- <html></html>: <html> элемент. Этот элемент оборачивает все содержимое страницы. Иногда его называют корневым элементом.
- <head></head>: <head> элемент. Этот элемент действует как контейнер для всего, что вы хотите включить в HTML-страницу, **а не для содержимого**, которое страница будет показывать зрителям. Сюда входят ключевые слова и описание страницы, которые будут отображаться в результатах поиска, CSS для стилизации контента, объявления наборов символов и многое другое. Подробнее об этом вы узнаете в следующей статье серии.
- <meta charset=«utf-8»>: <meta> элемент. Этот элемент представляет метаданные, которые не могут быть представлены другими мета-элементами HTML, такими как <base>,

<script>,<style>,<title> или <link>. Атрибут charset определяет кодировку символов вашего документа как UTF-8, которая включает большинство символов подавляющего большинства

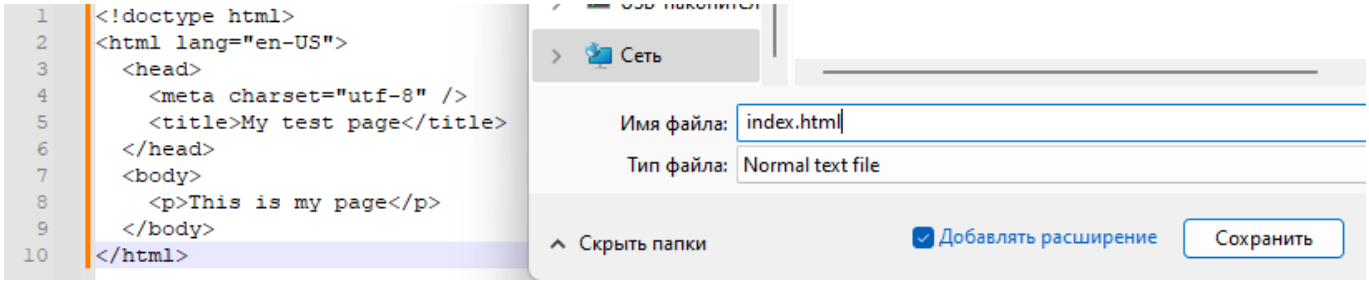
письменных языков, написанных человеком. Благодаря этому параметру страница теперь может обрабатывать любой текстовый контент, который она может содержать. Нет причин не устанавливать это значение, и это может помочь избежать некоторых проблем в дальнейшем.

- `<title></title>`: <title> элемент. Это устанавливает заголовок страницы, который отображается на вкладке браузера, в которую загружена страница. Заголовок страницы также используется для описания страницы, когда она добавлена в закладки.
- `<body></body>`: <body> элемент. Он содержит весь контент, отображаемый на странице, включая текст, изображения, видео, игры, воспроизводимые звуковые дорожки и что-либо еще.

Добавление функций в HTML-документ

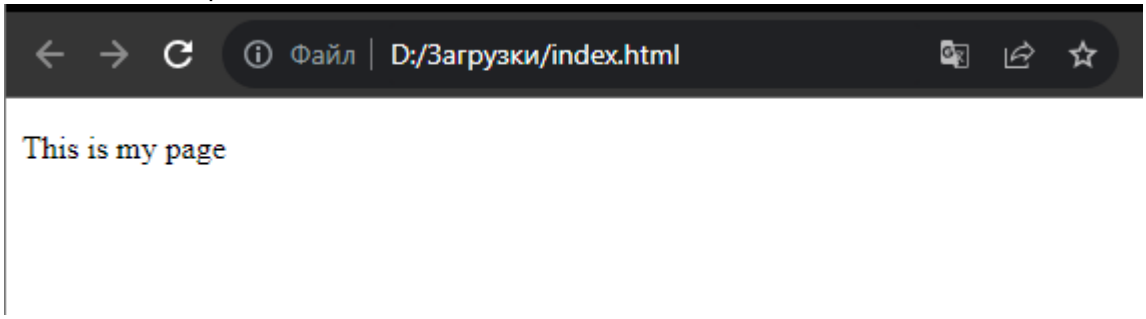
Если вы хотите поэкспериментировать с написанием HTML на локальном компьютере, вы можете:

1. Скопируйте приведенный выше пример HTML-страницы.
2. Создайте новый файл в текстовом редакторе.
3. Вставьте код в новый текстовый файл.
4. Сохраните файл как `index.html`.



Примечание. Этот базовый HTML-шаблон также можно найти в репозитарии.

Теперь вы можете открыть этот файл в веб-браузере и посмотреть, как выглядит визуализированный код. Отредактируйте код и обновите браузер, чтобы увидеть результат. Изначально страница выглядит так:



В этом упражнении вы можете редактировать код локально на своем компьютере, как описано ранее, или редактировать его в окне примера ниже (в данном случае редактируемое окно примера представляет только содержимое элемента <body>). Оттачивайте свои навыки, выполняя следующие задачи:

- Чуть ниже открывающего тега элемента <body> добавьте основной заголовок документа. Это должно быть заключено в <h1> открывающий и </h1> закрывающий теги.
- Отредактируйте содержимое абзаца, включив в него текст по теме, которая вам интересна.
- Выделите важные слова жирным шрифтом, заключая их в открывающий и закрывающий теги.
- Добавьте ссылку в свой абзац, как объяснялось ранее в статье .
- Добавьте изображение в документ. Разместите его под абзацем, как объяснялось ранее в статье . Заработайте бонусные баллы, если вам удастся создать ссылку на другое изображение (локально на вашем компьютере или где-то еще в Интернете).

Если вы допустили ошибку, вы всегда можете сбросить ее с помощью кнопки Reset.

Пробелы в HTML

В приведенных выше примерах вы могли заметить, что в коде содержится много пробелов. Это необязательно. Эти два фрагмента кода эквивалентны:

HTML

```
<p id="noWhitespace">Dogs are silly.</p>

<p id="whitespace">Kats

    are

        silly.</p>
```

Независимо от того, сколько пробелов вы используете внутри содержимого HTML-элемента (которое может включать один или несколько пробелов, а также разрывы строк), анализатор HTML уменьшает каждую последовательность пробелов до одного пробела при рендеринге кода. Так зачем использовать так много пробелов? Ответ: читабельность.

Будет легче понять, что происходит в вашем коде, если он будет правильно отформатирован. В нашем HTML каждый вложенный элемент имеет отступ на два пробела больше, чем тот, который находится внутри. Вы сами можете выбрать стиль форматирования (например, сколько пробелов для каждого уровня отступов), но вам следует подумать о его форматировании.

Давайте посмотрим, как браузер отображает два приведенных выше абзаца с пробелами и без них:

Примечание. При доступе к внутреннему коду HTML элементов из JavaScript все пробелы останутся нетронутыми. Это может привести к неожиданным результатам, если

```
пробелы обрезаны браузером.
```

JS

```
let a = document.getElementById("noWhitespace").innerHTML;
document.getElementById("demo_a").innerHTML = a;
// "Dogs are silly."
let b = document.getElementById("whitespace").innerHTML;
document.getElementById("demo_b").innerHTML = b;
// "Kats
//    are
//        silly."
```

Ссылки на сущности: включение специальных символов в HTML.

В HTML символы <, >, «, ' и &являются специальными символами. Они являются частью самого синтаксиса HTML. Так как же включить в текст один из этих специальных символов? Например, если вы хотите использовать амперсанд или знак «меньше» и не интерпретировать его как код.

Вы делаете это с помощью ссылок на персонажей. Это специальные коды, обозначающие символы, которые следует использовать именно в этих обстоятельствах. Каждая ссылка на символ начинается с амперсанда (&) и заканчивается точкой с запятой (;).

Символ	Эквивалент ссылки на символ
<	<
>	>
«	"
'	'
&	&

Эквивалент ссылки на символ можно легко запомнить, поскольку текст, который он использует, можно рассматривать как меньший, чем для '<', кавычка для ' » ' и то же самое для других. Дополнительные сведения о ссылках на сущности см. в разделе Список ссылок на символьные сущности XML и HTML (Википедия).

В примере ниже есть два абзаца:

HTML

```
<p>In HTML, you define a paragraph using the <p> element.</p>

<p>In HTML, you define a paragraph using the &lt;p&gt; element.</p>
```

На живом выводе ниже вы можете видеть, что первый абзац пошёл не так. Браузер интерпретирует второй экземпляр <p>как начало нового абзаца. Второй абзац выглядит нормально, поскольку в нем есть угловые скобки со ссылками на символы.

Примечание. Вам не нужно использовать ссылки на сущности для каких-либо других символов, поскольку современные браузеры прекрасно обрабатывают фактические символы, если для кодировки символов вашего HTML установлено значение UTF-8.

HTML-комментарии

В HTML есть механизм записи комментариев в коде. Браузеры игнорируют комментарии, фактически делая комментарии невидимыми для пользователя. Цель комментариев — позволить вам включать в код примечания для объяснения вашей логики или кода. Это очень полезно, если вы вернетесь к базе кода после того, как отсутствовали достаточно долго и не полностью ее помните. Аналогично, комментарии неоценимы, поскольку разные люди вносят изменения и обновления.

Чтобы написать комментарий HTML, оберните его специальными маркерами <!– и –>. Например:

HTML

```
<p>I'm not inside a comment</p>

<!-- <p>I am!</p> -->
```

Как вы можете видеть ниже, в реальном выводе отображается только первый абзац.

Заключение

Вы дочитали статью до конца! Мы надеемся, что вам понравился экскурс по основам HTML.

На этом этапе вы должны понимать, как выглядит HTML и как он работает на базовом уровне. Вы также должны уметь написать несколько элементов и атрибутов. Последующие статьи этого модуля развивают некоторые темы, представленные здесь, а также представляют другие концепции языка.

Когда вы начнете больше изучать HTML, подумайте об изучении основ CSS (каскадных таблиц стилей). CSS — это язык, используемый для стилизации веб-страниц, например изменения шрифтов или цветов или изменения макета страницы. Как вы вскоре обнаружите, HTML и CSS хорошо работают вместе.