Начало работы с HTML

В этой статье мы рассмотрим абсолютные основы HTML. Для начала в этой статье даны определения элементов, атрибутов и всех других важных терминов, которые вы, возможно, слышали. Также объясняется, где они вписываются в HTML. Вы узнаете, как структурированы элементы HTML, как структурирована типичная страница HTML и другие важные базовые функции языка. Попутно будет возможность поиграть и с HTML!

Что такое HTML?

HTML (язык разметки гипертекста) — это язык разметки , который сообщает веб-браузерам, как структурировать посещаемые вами веб-страницы. Это может быть настолько сложно или просто, насколько того хочет веб-разработчик. HTML состоит из ряда элементов, которые вы используете для включения, переноса или разметки различных частей контента, чтобы он выглядел или действовал определенным образом. Заключающие теги могут превращать контент в гиперссылку для перехода на другую страницу, выделять слова курсивом и т. д. Например, рассмотрим следующую строку текста:

```
My cat is very grumpy
```

Если бы мы хотели, чтобы текст оставался отдельным, мы могли бы указать, что это абзац, заключив его в <p> элемент абзаца ( ):

HTML

```
<p>My cat is very grumpy</p>
```

Примечание. Теги в HTML не чувствительны к регистру. Это означает, что их можно писать как прописными, так и строчными буквами. Например, <title> тег можно записать как <title>, <TITLE>, <Title>, <TiTlE>и т. д., и он будет работать. Однако лучше всего писать все теги строчными буквами для обеспечения единообразия и читабельности.

Анатомия HTML-элемента

Давайте подробнее изучим наш элемент абзаца из предыдущего раздела:

Анатомия нашей элемента такова:

- **Открывающий тег:** состоит из имени элемента (в данном примере p для абзаца), заключенного в открывающие и закрывающие угловые скобки. Этот открывающий тег отмечает начало или начало действия элемента. В этом примере он предшествует началу текста абзаца.
- **Содержимое:** это содержимое элемента. В данном примере это текст абзаца.
- **Закрывающий тег:** он аналогичен открывающему тегу, за исключением того, что он включает косую черту перед именем элемента. Это отмечает, где заканчивается элемент. Отсутствие закрывающего тега — распространенная ошибка новичков, которая может привести к необычным результатам.

Элемент представляет собой открывающий тег, за которым следует содержимое, а затем закрывающий тег.

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

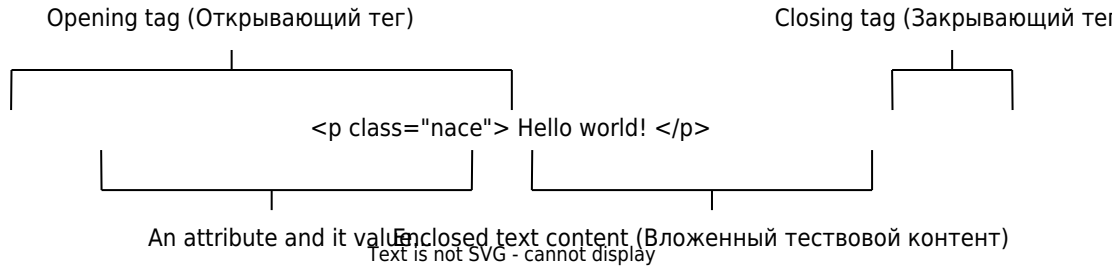
```
<img
  src="images/firefox-icon.png"
  alt="Firefox icon" />
```

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 " 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

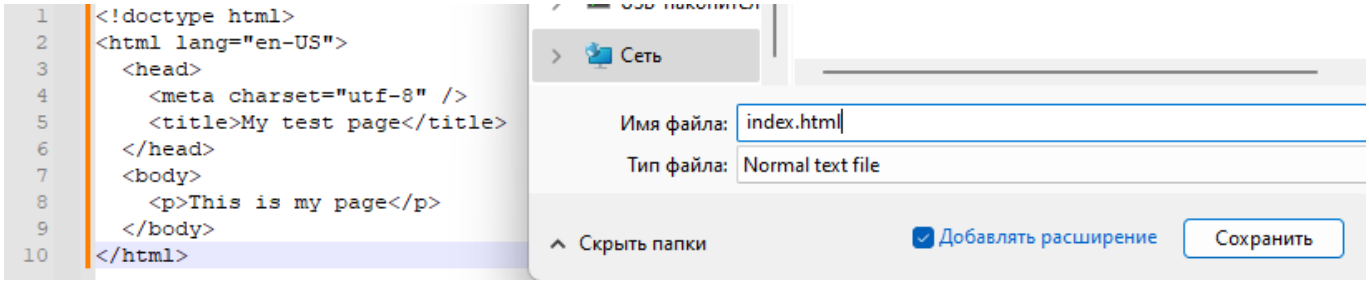
документа как UTF-8, которая включает большинство символов подавляющего большинства письменных языков, написанных человеком. Благодаря этому параметру страница теперь может обрабатывать любой текстовый контент, который она может содержать. Нет причин не устанавливать это значение, и это может помочь 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

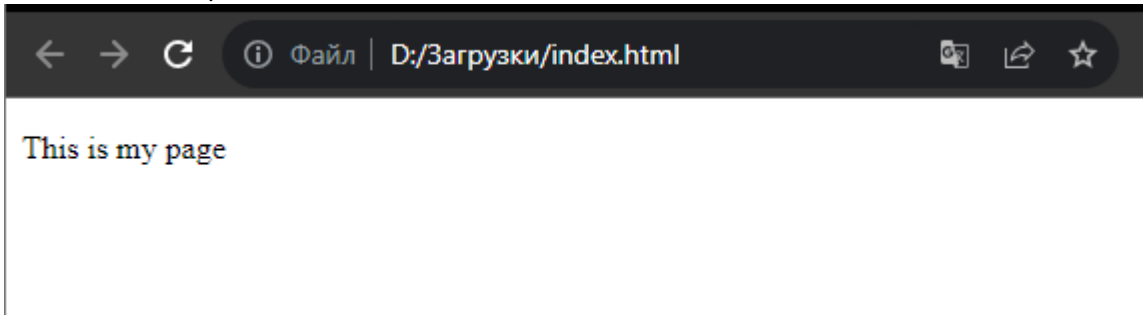
Если вы хотите поэкспериментировать с написанием HTML на локальном компьютере, вы можете:

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">Dogs

    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: