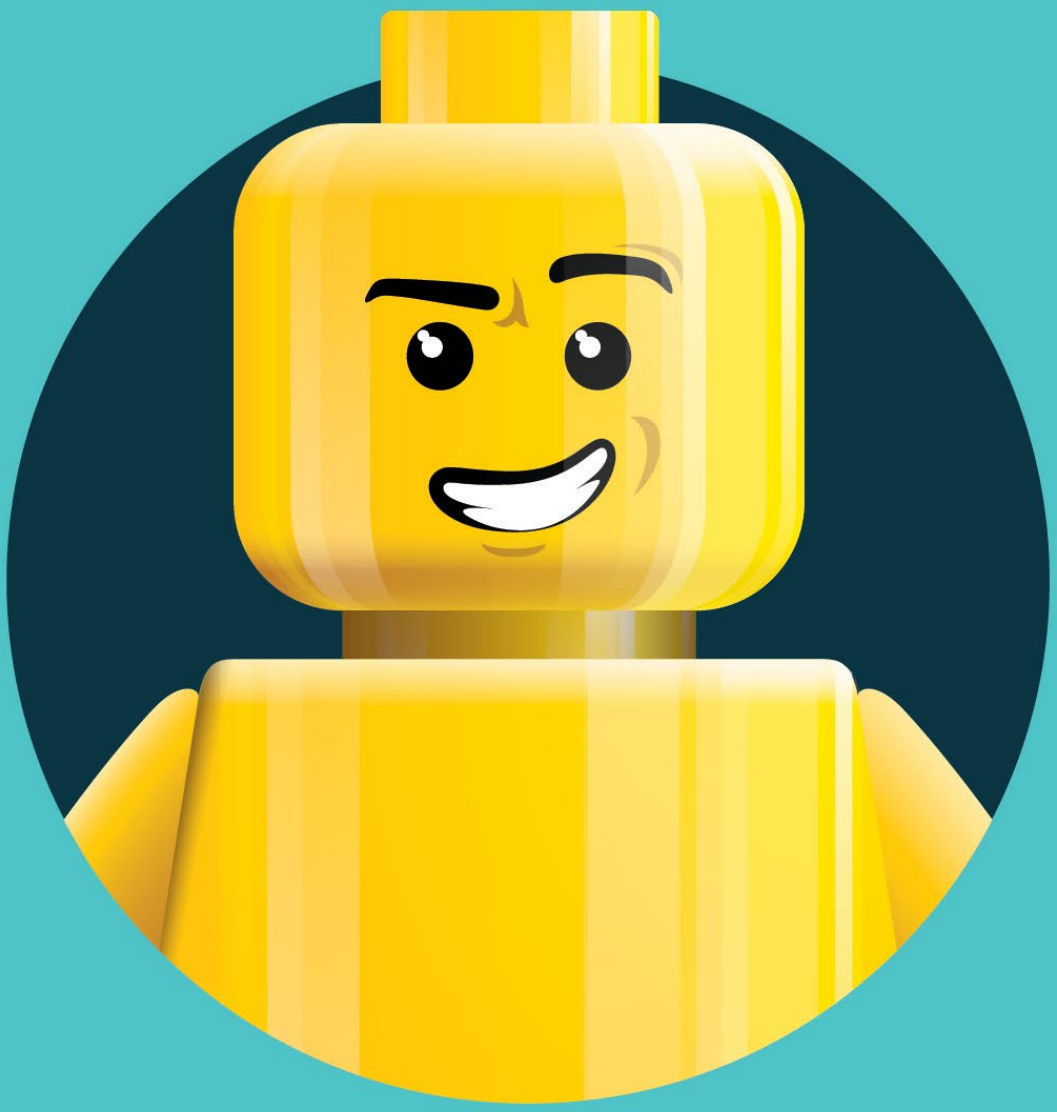

HTML

LoftSchool
от мыслителя к создателю

Содержание

1

Структура документа

HTML расшифровывается как HyperText Markup Language (в переводе означает Язык Разметки ГиперТекста).

HTML-документ, по сути, обычный текстовый файл, который содержит нормально читаемый текст и специальные команды разметки (tags или теги), заключенные в угловые скобки (**< и >**). Теги языка HTML задают правила, по которым браузер выводит документ на экран: размещение текста в окне, представление графических объектов (рисунков и картинок), а также вывод звуковых, видео клипов и т. д. HTML-теги обычно используются в паре, например: ****. Первый тег называется начальным тегом, а второй конечным тегом.

Текст, находящийся между начальным и конечным тегом, подвергается "разметке". Например **Привет** будет отображено браузером как **Привет** (слово 'Привет' написанное жирным шрифтом).

Следует иметь ввиду, что не все теги совместимы с браузерами. Если браузер не понимает тег, то он его просто пропускает.

Простейший HTML-документ имеет следующую структур:

```
<!DOCTYPE html >
<html>
   <head>
       <title>
           Название
       </title>
   </head>
   <body>
       Текст страницы
   </body>
</html>
```

Файл: index.html

Все HTML5-документы должны начинаться с объявления **DOCTYPE**. Предыдущие версии HTML имели несколько типов DOCTYPE. HTML5 имеет только один:

```
<!DOCTYPE html>
```

Данное объявление переводит все браузеры в нормальный режим. Браузеры не поддерживающие HTML5 в данном режиме будут интерпретировать старые теги и игнорировать новые, которые они не поддерживают.

Тег <html> является контейнером, который заключает в себе всё содержимое веб-страницы, включая теги **<head>** и **<body>**.

Тег <head> предназначен для хранения других элементов, цель которых — помочь браузеру в работе с данными. Также внутри контейнера **<head>** находятся метатеги, которые используются для хранения информации, предназначенной для браузеров и поисковых систем. Например, механизмы поисковых систем обращаются к метатегам для получения описания сайта, ключевых слов и других данных.

Элементы, располагающиеся в секции head не отображаются явно на странице и используются для служебных функций.

В секции **head** могут располагаться скрипты, инструкции об оформлении страницы и различная мета-информация о данном HTML-документе.

Метаданные – это информация о данных, находящихся в HTML-документе. Пример метаданных: кодировка страницы, краткое описание содержимого, ключевые слова, имя автора, дата последней модификации.

Метаданные не отображаются явно на странице, но используются браузерами и поисковыми системами.

В HTML метаданные HTML-документов определяются с помощью тега **<meta>**. ***Тэг <meta>*** всегда должен располагаться в секции **head.**

Для того, чтобы указать браузеру пользователя какая кодировка используется на данной странице, необходимо использовать атрибут **charset** тега meta например так

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

Если явно не указать кодировку, браузер, при отображении страницы, будет определять ее автоматически. Если кодировка при этом будет определена неверно, пользователь увидит страницу, содержащую бессмысленные символы, поэтому кодировка обязательно должна указываться к каждому HTML-документу.

Любой HTML-документ имеет название, заключенное между тегами **<title>** и **</title>**. По названию документа HTML браузеры могут найти информацию, поэтому место для названия всегда определено – оно находится вверху экрана и отдельно от содержимого документа. Максимальная длина названия – 40 символов. Пустой заголовок, не содержащий ни одного символа, включая пробел, не допустим. Также запрещено включать в код два и более элемента **<title>**, он должен быть только один.

Элемент title:

- Определяет заголовок окна браузера;
- Используется как заголовок страницы в результатах выдачи поисковых систем;
- Используется как заголовок страницы при добавлении сайта в избранное.

Тег <body> предназначен для хранения содержания веб-страницы, отображаемого в окне браузера. Информацию, которую следует выводить в документе, следует располагать именно внутри контейнера **<body>**. К такой информации относится текст, изображения, таблицы, списки и др. элементы. **HTML-элементом** называется комбинация начального тега, конечного тега и содержимого. Пример HTML-элемента:

```
<p> Это абзац. </p>
```

Большинство элементов могут быть вложены друг в друга (т.е. в содержимом одного элемента может располагаться другой элемент).

HTML не чувствителен к регистру. Это значит, что тег **<b>** будет интерпретироваться браузером так же, как и тег **<B>**. Несмотря на то, что HTML не настаивает на строгом синтаксисе, мы рекомендуем заранее выработать определенные правила написания кода (например, если вы пишете теги в нижней раскладке, то делайте так всегда).

HTML-элементы могут иметь атрибуты. Атрибуты задаются в начальном теге элементов и состоят из имени и значения. Например, в атрибуте **`href="https://loftschool.com/"`** **href** является именем, а **https://loftschool.com/** значением.

2

Текстовые абзацы

Изучим основные теги для логической разметки текста. Использовать их можно только внутри тега **<body>**.

Начнём с простейшего тега **<p>**, с помощью которого создаются абзацы. По умолчанию абзацы начинаются с новой строки и имеют вертикальные отступы, которыми можно управлять с помощью стилей.

```
<p> Я текстовый абзац </p>
```

Для выделения текста в HTML-документах могут быть использованы следующие теги: с помощью тега ****, **** можно установить жирный шрифт, тег ****, **** устанавливает наклонный шрифт. Однако, лучше для оформления всегда использовать таблицы стилей **CSS**.

Допускается также использование в тексте верхних и нижних индексов, соответственно, с помощью тегов **[,]** и **_,**.

С помощью HTML-тега **<pre>** вы можете отобразить предформатированный текст.

Все, что находится внутри тега **<pre>** будет отображено точно так, как написано. Браузер не будет удалять идущие подряд пробелы и символы перевода строки.

Установка цветов в HTML-документе производится с использованием таблицы стилей CSS.

Выбор цвета можно производить разными способами: например, заданием имени или определением RGB-номера выбранного цвета.

Поддерживаются следующие имена цветов: AQUA, BLACK, BLUE, FUCHSIA, GRAY, GREEN, LIME, MAROON, NAVY, OLIVE, PURPLE, RED, SILVER, TEAL, WHITE, YELLOW.

Номер цвета RGB задается тремя двухзначными шестнадцатеричными числами, причем каждое число из интервала 00-FF и определяет интенсивность соответствующего цвета. Например, номер цвета #FF0000 соответствует красному цвету, так как имеет максимальную интенсивность для красного, а зеленый и голубой имеют значения, равные нулю.

Соответственно, номер #00FF00 кодирует зеленый цвет и номер #0000FF – голубой.

После введения CSS некоторые теги и атрибуты в HTML стали считаться устаревшими (с помощью CSS можно добиться большего и гораздо эффективнее).

Избегайте использования следующих устаревших HTML-тегов:

`<strike>`	Определяет зачеркнутый текст.
`<font> и <basefont>`	Устанавливает шрифт для текста.
`<center>`	Выравнивает содержимое по центру.
`<menu>`	Определяет список меню.

В HTML5 появился новый тег **<mark>**, который обозначает выделенный текст. Иногда при работе с объёмными текстами мы используем маркер, чтобы выделять ключевые слова, идеи или что-то другое, на что стоит обратить внимание. Такое же назначение и у тега **<mark>**. В современных браузерах текст внутри **<mark>** подсвечивается жёлтым фоном.

3

Заголовки и подзаголовки

Для создания структуры больших текстов обычно используются заголовки. В текстовых редакторах есть возможность выделить часть текста, найти пункт «Заголовок» нужного уровня в меню, и применить его.

Прежде чем остаться на странице и приступить к ее прочтению, пользователи, обычно, бегло пробегают глазами по ее содержимому, проверяя, содержит ли она интересующую их информацию. Заголовки – это первое (и часто единственное) на что они обращают внимание, поэтому неправильное использование заголовков может привести к потере посетителей.

В первую очередь, заголовки должны кратко и точно описывать содержимое которое они озаглавливают. Наиболее важная информация страницы должна располагаться под заголовками большего размера, а наименее важная – под заголовками меньшего размера.

В языке HTML для выделения заголовков предусмотрено целое семейство тегов: от **<h1>** до **<h6>**. Тег **<h1>** обозначает самый важный заголовок (заголовок верхнего уровня), а тег **<h6>** обозначает подзаголовок самого нижнего уровня.

На практике редко встречаются тексты, в которых встречаются подзаголовки ниже третьего уровня. Поэтому самыми часто используемыми тегами заголовков являются: **<h1>**, **<h2>** и **<h3>**.

Стоит отметить, что поисковые системы придают особое значение заголовкам, поэтому необходимо учиться правильно их использовать.

4

Ссылки

Важнейшим свойством HTML являются ссылки, позволяющие связать текст или картинку с другими гипертекстовыми документами. Текст, как правило, выделяется цветом и оформляется подчеркиванием, для чего используется тег **<a>**:

```
<a href = ”filename”>текст_ссылки</a>
```

filename — имя файла или адрес Internet, на который необходимо сослаться, а **текст_ссылки** — текст гипертекстовой ссылки, который будет непосредственно показан в HTML-документе. Например, гипертекстовая ссылка

```
<a href = ”my_work.html”>Views</a>
```

ссылается на документ **my_work.html**, образуя гипертекстовую ссылку в виде слова **Views**. Если документ, формирующий ссылку, находится в другой папке, относящейся к web-сайту, то подобная ссылка называется относительной. Например:

```
<a href = ”photo/my_photo.html”>Мой фотоальбом</a>
```

ссылается на файл **my_photo.html**, расположенный в папке photo, вложенной в текущую, и образует ссылку в виде текста **Мой фотоальбом**. Если есть необходимость сослаться на ресурс Internet, расположенный на удаленном сервере, или указать в ссылке полное имя файла и путь к файлу, то используют абсолютные ссылки. Структура такого тега аналогична, только он формируется на основе полного пути к ресурсу в виде **протокол:/URL/путь**. Например:

```
<a href=”http://www.yandex.ru”>Поиск в интернет</a>
```

В HTML гиперссылки делят на два вида:

- **Внешние гиперссылки** перемещают пользователя, кликнувшего на них в другой HTML-документ. Как те, которые мы рассмотрели выше.
- **Внутренние гиперссылки** (якоря) перемещают пользователя на предварительно созданную закладку в документе, в котором они определены.

```
<!-- Создание гиперссылки на закладку -->
<a href="#bookmark"> Текст ссылки </a>
<!-- Создание закладки -->
<div id="bookmark"> Текст закладки. </div>
```

5

Изображение

В документах HTML могут использоваться изображения и графика, для чего используется тег **<img>**. Допускается использование файлов в формате **PNG**, **GIF** или **JPG/JPEG**. Следующий пример демонстрирует вставку в документ JPG -файла:

```
<img src="pic.jpg" width=45 height=50 alt="Photo">
```

Здесь атрибут **src=** определяет URL-адрес графического файла. В приведенном примере файл будет размещен в области шириной 45 и высотой 50 пикселей соответственно. Если размеры, указанные атрибутами **height= (высота)** и **width= (ширина)**, не совпадают с размерами графического файла, то последний масштабируется. Для графических файлов рекомендуется всегда задавать размеры в таблицах стилей CSS вместо атрибутов **height** и **width**.

Атрибут **alt=** указывает на то, что подставить вместо рисунка, если браузер не показывает графические файлы или, вследствие медленной скорости соединения, файл еще не получен. Данный атрибут обязателен с точки зрения семантики и валидатор будет выдавать ошибку при его отсутствии. Роботы поисковых систем, которые анализируют страницы, не могут просматривать картинки, поэтому единственный способ "рассказать" им о содержимом – написать об этом в атрибуте **alt**.

Картинка может быть средством задания и управления выбором гиперссылок в HTML-документе, для чего на тег img должна указывать гиперссылка, определяемая тегом a. Например:

```
<a href ="index.htm">
   <img src="logo.jpg" width= "43" height="35" alt="На
   главную страницу">
</a>
```

6

Списки

Язык HTML имеет возможности для создания различных списков и перечислений. Для их создания могут использоваться теги **ul** и **ol**, а элементы списка отмечаются тегом **li**, при этом допускаются вложенные списки любой глубины.

Рассмотрим следующий пример ненумерованного списка и использования тегов **ul** и **li**

```
<ul>
   <li> красный</li>
   <li> оранжевый </li>
   <li> желтый </li>
   <li> зеленый </li>
   <li> голубой </li>
   <li> синий </li>
   <li> фиолетовый </li>
</ul>
```

что даст в окне браузера результат, показанный на рис. 1.

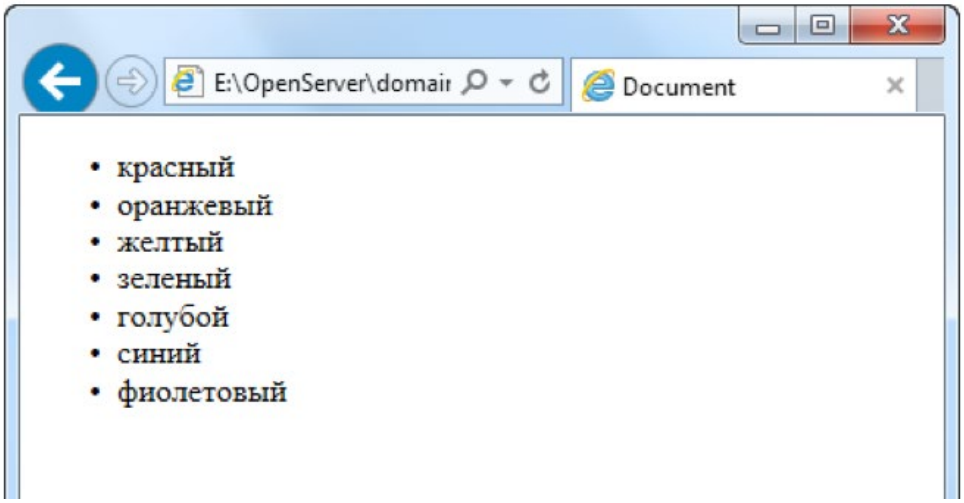

Рис. 1. Пример ненумерованного списка

Нумерованный список создают, используя теги **ol** и **li** Например:

```
<ol>
   <li>элемент первый.
   <li>элемент второй.
   <li>элемент третий.
</ol>
```

По умолчанию нумерация дается арабскими цифрами, начиная с единицы. Используя атрибуты тега ol, можно изменить стиль оформления списка. Атрибут **type=** определяет стиль нумерации (буквы или цифры), при этом допускаются следующие его значения:

`type = A`	использовать большие латинские буквы
`type = a`	использовать маленькие буквы
`type = I`	использовать большие римские цифры
`type = i`	использовать маленькие римские цифры
`type = 1`	использовать арабские цифры

Атрибут **start=** определяет начальное значение списка (десятичное число), например **start=5**.

И, наконец, рассмотрим пример вложенного списка, в котором допускается использовать теги **ul**, **ol** и **li**

```
<ul>
<li>Первый раздел </li>
   <ul>
        <li>Первый подраздел первого раздела </li>
        <li>Второй подраздел первого раздела
            <ol type = “a”>
                <li>Первый список </li>
                <li>Второй список </li>
            </ol>
</li>
        <li>Третий подраздел первого раздела </li>
   </ul>
<li>Второй раздел </li>
</ul>
```

который в браузере будет выглядеть так, как показано на рис. 2.

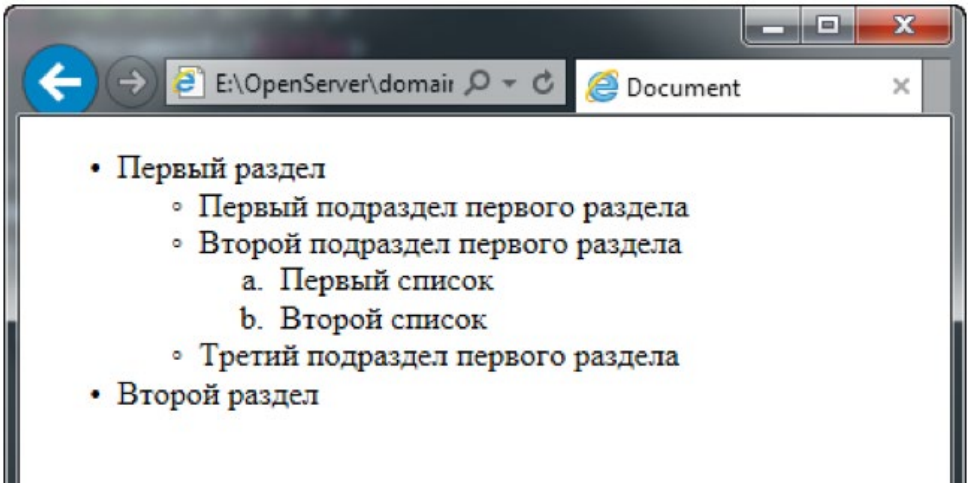

Рис. 2. Пример вложенного списка

7

Комментарии

Тег добавляет комментарий в код документа. Текст комментария не отображается на странице. Разрешается внутрь комментария добавлять другие теги, вложенные комментарии (когда один комментарий расположен внутри другого) недопустимы.

```
<!—текст комментария ——>
```

С помощью комментариев вы можете оставлять в коде пояснительные заметки.

- Комментарии полностью игнорируются браузером.
- Комментарии не отображаются при просмотре страницы.

Комментарии могут использоваться при отладке кода. Если вы не уверены, хотите ли видеть данный заголовок, абзац и т.д. в итоговой версии страницы, оберните код в тэг **<!-- >** на время принятия решения.

8

Спецсимволы HTML

Помимо тегов в языке HTML используется специальный управляющий символ **&** — амперсант. Этот символ используется для вывода специальных символов и символов из расширенной кодовой таблицы, которые нельзя ввести с клавиатуры. Например, вывод самого символа амперсант **&** осуществляется посредством последовательности символов **&** для вывода угловых скобок используются **<** для "**<" и > для ">**", а, скажем, символ с номером **182** из кодовой таблицы, может быть задан последовательностью **¶**.

Самый часто используемый спецсимвол – это неразрывный пробел (его мнемоника ** **).

Данный символ используется для контролирования переноса строки (после данного символа автоматический перевод строки невозможен) и для вставки в текст подряд идущих пробелов (по умолчанию, если вы вставите в текст 5 подряд идущих пробелов, браузер вырежет 4 и отобразит лишь один).
Мнемоники и коды часто используемых спецсимволов

Символ	Описание	Мнемоника	Код
	Пробел		
<	Меньше чем	&lt;	&#60;
>	Больше чем	&gt;	&#62;
©	Знак охраны авторского права	&copy;	&#169;
®	Зарегистрированный товарный знак	&reg;	&#174;
&	Знак амперсанда	&amp;	&#38;

9

Семантические теги

Благодаря использованию CSS вы можете создавать страницы с хорошо понятной для пользователей визуальной структурой, но будут ли эти страницы также понятны для поисковых систем или браузеров?

Например, как поисковый робот может отличить содержимое документа от навигационного меню если они размечены с помощью одинаковых div-элементов?

Для того, чтобы разрешить эту проблему, в HTML5 были введены семантические теги. С помощью семантических тегов вы можете сделать страницы сайтов более понятными для поисковых систем и браузеров.

`<footer>`	Определяет футер
`<header>`	Определяет заголовочный блок сайта
`<nav>`	Определяет навигационное меню

С помощью тега **<section>** вы можете группировать логически связанное содержимое в документе.

Если логически связанное содержимое является автономным (может использоваться в других документах независимо от остального содержимого на странице) необходимо использовать вместо **<section>** тег **<article>**.

Тег **aside** используется для выделения элементов, которые не являются частью содержимого, но косвенно с ним связаны.

Данным тегом могут выделяться: цитаты, дополнительная информация к статье, словарь с терминами, список ссылок и т.д.

Как уже говорилось выше, с помощью тега **<mark>** вы можете выделить "важную" часть в тексте.

В журналах и газетах иллюстрации часто сопровождаются подписями. В HTML4 невозможно было создавать подписи, не прибегая к использованию CSS.

В HTML5 это проблема решена добавлением новых тегов: **<figure>** и **<figcaption>**.

```
<figure>
   <img src='foto-1387634.jpg' width='300' height='230' />
   <figcaption>Мы с Машей на отдыхе, Красное море </figcaption>
</figure>
```

10

Форма

Функциональность сайтов, предоставляющих пользователю возможность ввода данных и получения результатов их обработки, обеспечивается использованием программ, работающих на стороне сервера — серверных приложений. Эти приложения обрабатывают полученные от посетителя Web-сайта данные и выдают результат в виде обычной Web-страницы. Именно для них в HTML предусмотрена возможность создания Web-форм и **элементов управления** — чтобы посетитель мог ввести данные, которые потом обработает **серверное приложение**.

Основная схема работы серверного приложения:

1. Посетитель вводит в элементы управления, расположенные в Web-форме на Web-странице, нужные данные.

2. Введя данные, посетитель нажимает расположенную в той же Web-форме особую кнопку — кнопку отправки данных.

3. Web-форма кодирует введенные в нее данные и отправляет их серверному приложению, расположенному по указанному интернет-адресу.

4. Web-сервер перехватывает отправленные данные, запускает серверное приложении и передает данные ему.

5. Серверное приложение обрабатывает полученные данные.

6. Серверное приложение формирует ответ (возможно, это Web-страница) с результатами обработки данных посетителя и передает ее Web-серверу.

7. Web-сервер получает сформированную серверным приложением ответ и отправляет ее посетителю.

Форма предназначена для обмена данными между пользователем и сервером. Область применения форм не ограничена отправкой данных на сервер, с помощью клиентских скриптов можно получить доступ к любому элементу формы, изменять его и применять по своему усмотрению.
Для создания интерактивных HTML-документов используют заполняемые формы, определяемые тегом **form**. В одном документе может быть определено несколько форм для заполнения, но одновременно на сервер может быть отправлена только одна форма. По этой причине данные форм должны быть независимы друг от друга и вложенные теги form не допускаются. Формат тега form следующий:

```
<form action="url" method="GET/POST"> ... </form>
```

Здесь используются следующие атрибуты:

- ***action=url*** – URL-адрес сервера запросов, куда будет отослано содержание формы после подтверждения. Если это поле отсутствует, будет использован URL-адрес текущего документа.

- ***method=GET/POST*** – HTTP-метод, используемый для посылки содержания заполненной формы на сервер. Возможные варианты при этом следующие:

  - ***GET*** – это метод по умолчанию, который приводит к добавлению содержимого заполненной формы к URL;

  - ***POST*** – при использовании этого метода содержимое заполненной формы пересылается не как часть URL, а как содержимое тела запроса.

  - ***ENCTYPE*** - задаёт тип кодирования содержимого заполненной формы. Этот атрибут действует только когда используется метод POST.

Свойства и методы формы:

Свойство	Описание
name	Имя формы
action	Адрес файла серверного сценария, который будет обрабатывать заполненную и переданную форму
method	Метод передачи данных серверу
encoding	Тип содержимого, используемый для отправки формы на сервер
target	Имя окна или фрейма для загрузки документа, сгенерированного сценарием на основании принятых из формы данных
length	Количество элементов формы
elements[]	Массив элементов формы

Когда форма отправляется на сервер, управление данными передаётся программе, заданной атрибутом **action** тега **<form>**. Предварительно браузер подготавливает информацию в виде пары **"имя=значение"**, где имя определяется атрибутом **name** тега **<input>**, а значение введено пользователем или установлено в поле формы по умолчанию.
Внутрь контейнера **<form>** помещаются другие теги, при этом сама форма никак не отображается на веб-странице, видны только ее элементы и результаты вложенных тегов.

11

Теги формы

Внутри тега **form** могут находиться любые теги, кроме другого тега **form**. Для задания интерфейсных элементов внутри form используются теги **input**, **select**, **textarea** и другие. Давайте их рассмотрим.

Для того чтобы ввести информацию о пользователе (логин, пароль, дату рождения и т.д.) используют элемент ввода **<input>**. Его основной атрибут – **type** – определяет, что мы будем вводить и как.

Тег **input** используется для задания простого элемента ввода, при этом могут быть использованы следующие атрибуты:

accept	устанавливает фильтр на типы файлов, которые вы можете отправить через поле загрузки файлов;
accesskey	задает комбинацию клавиш, позволяющую перейти к элементу;
align	определяет выравнивание изображения;
alt	альтернативный текст для кнопки с изображением;
autocomplete	включает или отключает автозаполнение, соответственно принимает значения on и off;
border	задает толщину рамки вокруг изображения.;
checked	предварительно активированный переключатель или флажок;
disabled	блокирует доступ и изменение элемента;
form	связывает поле с формой по её идентификатору;
formaction	определяет адрес обработчика формы;
formenctype	устанавливает способ кодирования данных формы при их отправке на сервер;
formmethod	сообщает браузеру каким методом следует передавать данные формы на сервер (GET или POST);

formnovalidate	отменяет встроенную проверку данных на корректность введенных значений;
formtarget	определяет окно или фрейм в которое будет загружаться результат, возвращаемый обработчиком формы;
list	указывает на список вариантов, которые можно выбирать при вводе текста;
max	максимально допустимое значение для ввода числа или даты;
maxlength	максимально допустимое количество символов разрешенных в тексте;
min	минимально допустимое значение для ввода числа или даты;
multiple	позволяет загрузить несколько файлов одновременно;
name	задает имя поля, предназначено для того, чтобы обработчик формы мог его идентифицировать;
pattern	устанавливает шаблон ввода;
placeholder	выводит подсказывающий текст;
readonly	устанавливает, что поле не может изменяться пользователем;
required	отмечает, что поле является обязательным для заполнения;
size	задает ширину текстового поля;
src	определяет адрес графического файла для поля с изображением;
step	задает шаг приращения для числовых полей;
tabindex	определяет порядок перехода между элементами с помощью клавиши Tab;
type	сообщает браузеру, к какому типу относится элемент формы.
value	определяет значение элемента.

Значения предпоследнего атрибута type в списке очень обширны и важны, поскольку задают поведение при вводе в **input**

Значение	Описание
button	Кнопка.
checkbox	Флажки. Позволяют выбрать более одного варианта из предложенных.
file	Поле для ввода имени файла, который пересылается на сервер.
hidden	Скрытое поле. Оно никак не отображается на веб-странице.
image	Поле с изображением. При нажатии на рисунок данные формы отправляются на сервер.
password	Обычное текстовое поле, но отличается от него тем, что все символы показываются звездочками.
radio	Переключатели. Используются, когда следует выбрать один вариант из нескольких предложенных.
reset	Кнопка для возвращения данных формы в первоначальное значение.
submit	Кнопка для отправки данных формы на сервер.
text	Текстовое поле. Предназначено для ввода символов с помощью клавиатуры.
color	Виджет для выбора цвета.
date	Поле для выбора календарной даты.
datetime	Указание даты и времени.
datetime-local	Указание местной даты и времени.
email	Для адресов электронной почты.

number	Поле ввода чисел. Ввод символов, не являющихся цифрами, приведет к выводу предупреждения.
range	Ползунок для выбора чисел в указанном диапазоне.
search	Поле для поиска.
tel	Поле для телефонных номеров.
time	Поле для времени.
url	Поле для веб-адресов.
month	Поле для выбора месяца.
week	Поле для выбора недели.

Очень часто мы сталкиваемся с необходимостью выбора. Например, мы выбираем, согласны ли с лицензионным соглашением, мы выбираем какие странички отображать в меню соц. сети, приложения.

Когда надо выбрать несколько элементов из многих, используют **checkbox** – поле с квадратиком, куда устанавливается галочка. Создается это поле так:

```
<input id="Checkbox1" type="checkbox" value="Английский"/>
Английский<br />
```

value – значение, которое вы выбрали. Используется для обработки в скриптах. Рядом с полем пишется то слово, которое соответствует элементу выбора, и которое мы хотим отобразить на странице в браузере.

Иногда мы можем выбирать только один элемент из списка предложенных альтернатив. Например, мы выбираем пол: мужской/женский. Такой выбор нам помогут организовать радиокнопки.

Создаются они так:

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

Обратите внимание, что имена (name) – одинаковые. Это обеспечивает то, что вы сможете выбрать (одновременно) только один из элементов.

Иногда очень удобно задавать альтернативы в виде выпадающего списка. На некоторых сайтах так организован выбор страны, вуза, возраста. Задание перечня альтернатив удобно для обработки, так как пользователь правильно введёт данные (ну хотя бы с точки зрения орфографии). Использование выпадающего списка значительно экономит место на форме.

Тег **select** предназначен для создания списков в форме, при этом внутри разрешена только последовательность тегов **option**, за каждым из которых следует некоторое количество простого текста. Атрибуты тега **select** следующие:

name=идентификатор – символьное имя для элемента select, по которому он идентифицируется;

size=n – если значение равно 1 или если этот атрибут опущен, то элемент select будет представлен как выпадающее меню. Если size = 2 или более, то элемент будет представлен как окно выбора, а значение будет определять, сколько элементов списка будут видны;

multiple – если этот атрибут присутствует, то допускается множественный выбор из списка.

Создать список можно так:

```
<select id="Select1">
<option>Элемент списка</option>
...
</select>
```

где **select** – сам список, **option** – элемент списка.

Можно так же организовать выбор из заданного списка: **datalist**. Это называется альтернативой и ее можно задавать с помощью списка вариантов:

```
Любимое блюдо

<input type="text" name="team" id="favorite_dish"
list="dish_list">
<datalist id="dish_list">
<option>Борщ</option>
<option>Суп</option>
<option>Щи</option>
<option>Окрошка</option>
</datalist>
```

Обратите внимание на то, что атрибут **list** элемента **input** и атрибут **id** элемента **<datalist>** содержат одинаковое значение dish_list. Это их связывает.

Тег **button** создает на веб-странице кнопки и по своему действию напоминает результат, получаемый с помощью тега **<input>** (с атрибутом **type="button | reset | submit"**). В отличие от этого тега, **<button>** предлагает расширенные возможности по созданию кнопок. Например, на

подобной кнопке можно размещать любые элементы HTML, в том числе изображения. Используя стили, можно определить вид кнопки путём изменения шрифта, цвета фона, размеров и других параметров.

Теоретически, тег **<button>** должен располагаться внутри формы, устанавливаемой элементом **<form>**. Тем не менее, браузеры не выводят сообщение об ошибке и корректно работают с тегом **<button>**, если он встречается самостоятельно. Однако, если результат нажатия на кнопку необходимо отправить на сервер, помещать **<button>** в контейнер **<form>** обязательно.

И, наконец, тег **textarea** может быть использован для расположения многострокового поля ввода с необязательным содержимым в форме.

Атрибуты тега textarea следующие:

name=	символьное имя поля ввода;
rows=	число строк в поле ввода, то есть высота поля;
cols=	число столбцов в поле ввода, то есть ширина поля.

Объект **textarea** имеет полосы прокрутки, так что может быть введено любое количество текста. Содержание по умолчанию должно быть строгим ASCII-текстом, при этом символы перевода строки воспринимаются.

12

Пример несложной формы

```
<form method="POST" id= "student"  name= "student" action =
'ajax.php'>
   <label for = "firstName"> Введите Ваше имя
        <input  type= "text" id = " firstName" name="firstName"
        size="45">
   </label>
   <input type="radio" checked name="radioJob">
        <span>Учащийся ВУЗа</span>
   <input type="radio"  name="radioJob" > <span>Другое</span>
   <button type="submit"> Submit </button>
   <button type= "reset"> Reset </button>
</form>
```

которая в браузере будет выглядеть так, как показано на рис. 1 .

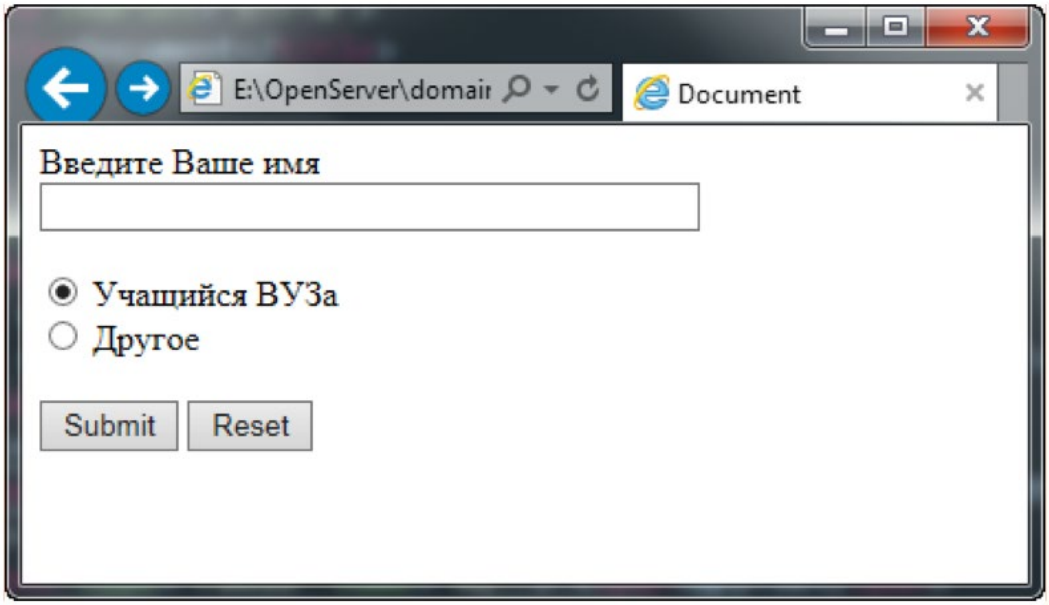

Рис. 1

13

Таблицы

В таблице каждый кусок информации отображается в ячейке (**cell**).
Ячейки, лежащие на одной линии, составляют строку (**row**). В некоторыех таблицах, строки могут быть сгруппированы. Специальная группа строк в начале таблицы – заголовок (**header**), в конце – нижний колонтитул (**footer**). Главные строки таблицы – тело (**body**), и они могут быть также сгруппированы.
Ячейки в линии сверху вниз, составляют столбец (**column**), но столбцы имеют ограниченное приминение в таблицах CSS.

Описание таблиц в HTML-документах осуществляется с помощью тега **<table>**. Таблица задается двумя тегами: **<tr>, </tr>** – описание строки таблицы и **<td>, </td>** – описание клетки таблицы.

Ячейки не имеют границ. У ячеек нет рамок и наполнений. По умолчанию, рамки разделены значениями таблицы, это свойство **border-spacing**.
Вы можете также полностью удалить пространство, установив свойство таблицы **border-collapse** в **collapse**.

По умолчанию, текст внутри таблицы выравнивается по левому краю, а ширина столбца таблицы определяется наиболее длинным элементом в этом столбце.

Допускается добавлять к таблице, строке или столбцу заголовок. Тег **caption** после тега **table** задает заголовок к таблице, который по умолчанию центрируется относительно таблицы и, по умолчанию, она отображается вверху таблицы.

Чтобы переместить её вниз, установите её свойство caption-side в bottom. Это свойство наследуется, поэтому, в качестве альтернативы вы можете установить это свойство у таблицы или у другого элемента предка. Чтобы стилизовать заголовок текста, используйте любое из обычных свойств для текста.

Тег **<caption>** должен размещаться внутри тега **<table>**, причём непосредственно внутри него и первым, до остальных вложенных тегов. Задание заголовка к строке или столбцу таблицы осуществляется при помощи тега **th** после tr или td соответственно.

Атрибут **border=** к тегу **table** рисует рамку вокруг таблицы и каждой клетки, при этом ширина рамки задается в пикселях. Следует иметь ввиду, что атрибуты **colspan=** и **rowspan=** тегов **td** и **tr** позволяют объединять клетки таблицы в группы, вокруг которых рисуется рамка.

Рассмотрим пример таблицы, занимающей по ширине весь экран браузера:

```
<table border="l">
   <tr>
       <td colspan="2">
       Две ячейки, объединенные по горизонтали
       </td>
   </tr>
   <tr>
       <td rowspan="2">
       Две ячейки, объединенные по вертикали
       </td>
       <td >просто ячейка</td>
   </tr>
   <tr>
       <td > просто ячейка </td>
   </tr>
</table>
```

что в окне браузера даст результат, приведённый на рис. 2.

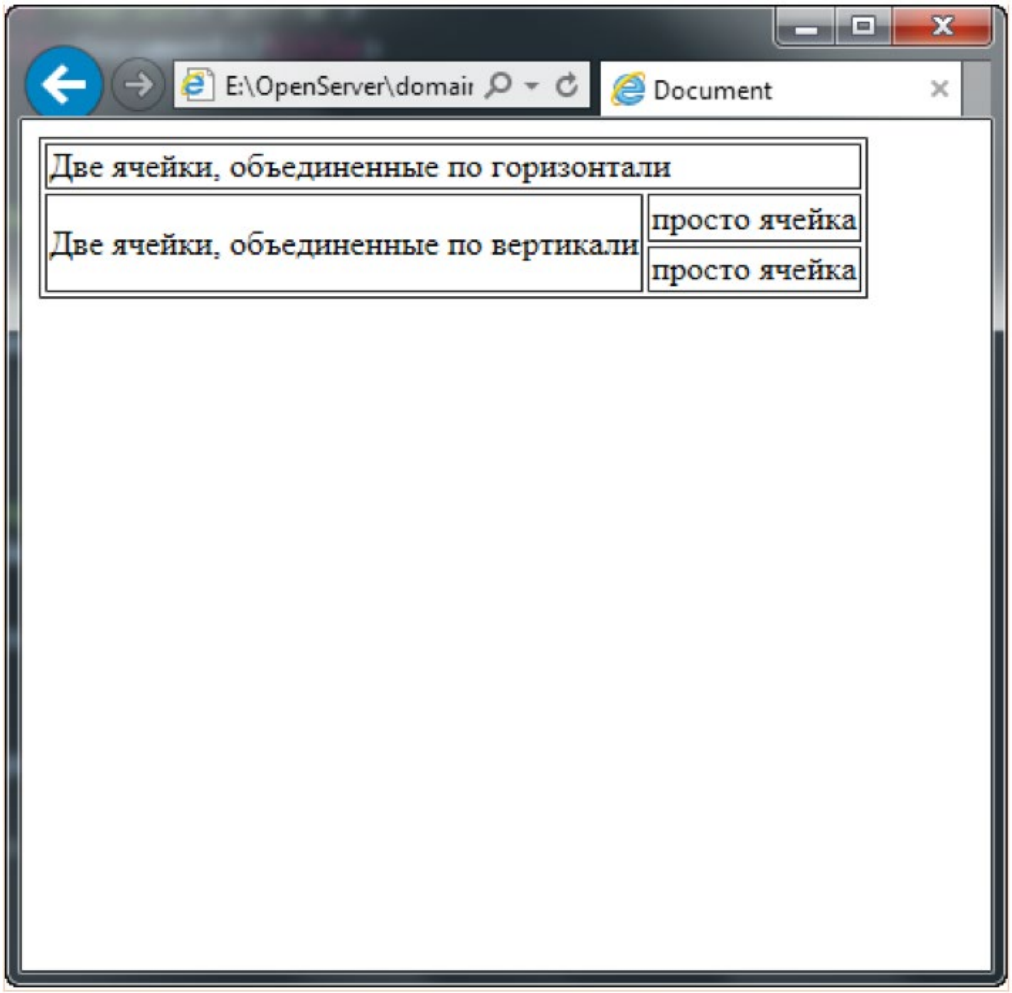

Рис. 2. Пример таблицы

Как хорошо стилизовать таблицу с помощью CSS? Необходимо знать несколько основных свойств и то, как они работают. Это уже упоминавшиеся **border-spacing** и **border-collapse.**

border-collapse устанавливает, как отображать границы вокруг ячеек таблицы. Это свойство играет роль, когда для ячеек установлена рамка, тогда в месте стыка ячеек получится линия двойной толщины. Значение **collapse** заставляет браузер анализировать подобные места в таблице и убирать в ней двойные линии. При этом между ячейками остаётся только одна граница, одновременно принадлежащая обеим ячейкам. То же правило соблюдается и для внешних границ, когда вокруг самой таблицы добавляется рамка. Значение по умолчанию **separate**.

Синтаксис:

```
border-collapse: collapse | separate
```

Значения: **collapse** – линия между ячейками отображается только одна, также игнорируется значение атрибута **cellspacing**.

separate – вокруг каждой ячейки отображается своя собственная рамка, в местах соприкосновения ячеек показываются сразу две линии.

border-spacing - задаёт расстояние между границами ячеек в таблице. **border-spacing** не работает в случае, когда для таблицы установлено свойство **border-collapse** со значением collapse. Значение по умолчанию **0**.

Синтаксис:

```
border-spacing: <размер> [<размер>]
```

Значения: Одно значение устанавливает одновременно расстояние по вертикали и горизонтали между границами ячеек. Если значений два, то первое определяет горизонтальное расстояние, а второе — вертикальное.